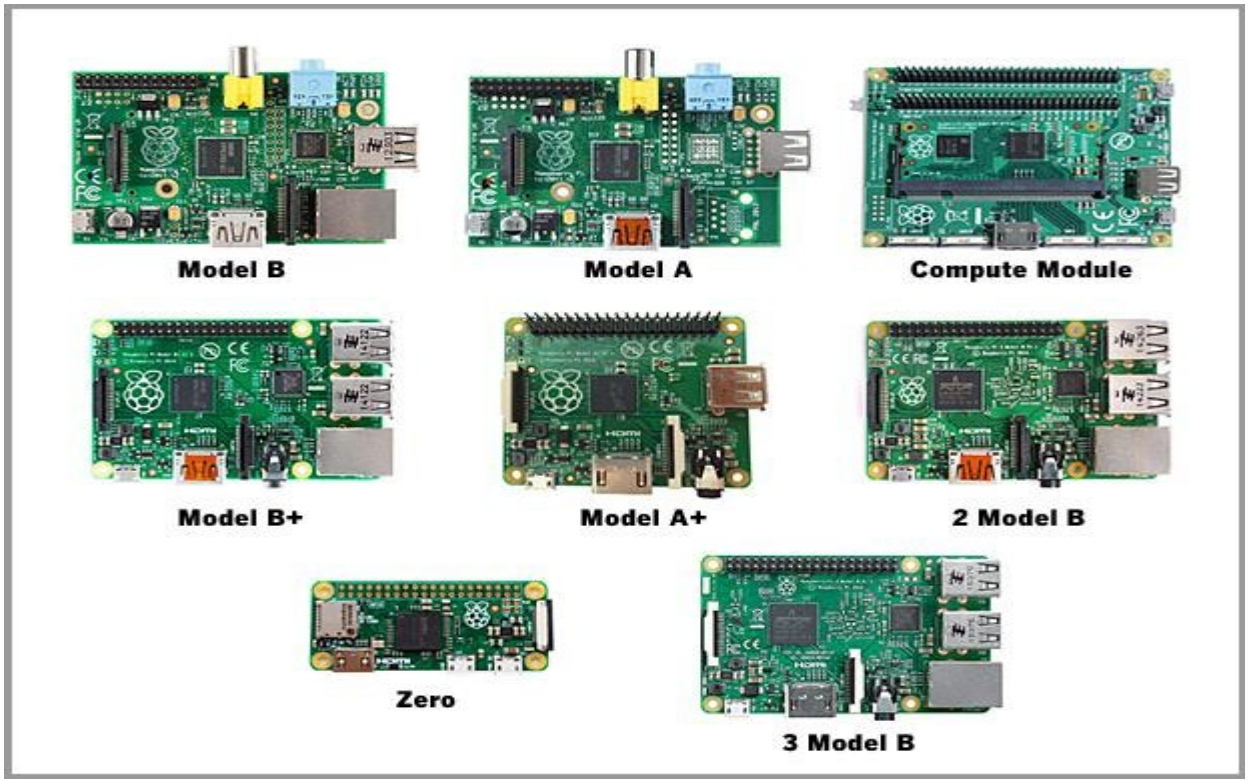
100 BMOC Raspberry Pi and IoT Curriculum



Rpi Board Features Summary

Raspberry Pi Boards Feature Summary

	Model B	Model A	Compute Module	Model B+	Model A+	2 Model B	Zero	3 Model B
Release	Apr. 2012	Feb. 2013	Apr. 2014	Jul. 2014	Nov. 2014	Feb. 2015	Nov. 2015	Feb. 2016
RAM	512MB	256MB	512MB			1GB	512MB	1GB
System on a Chip	Broadcom BCM2835					Broadcom BCM2836	Broadcom BCM2835	Broadcom BCM2837
Processor	ARMv6 (32-bit)					ARMv7 (32-bit)	ARMv6 (32-bit)	ARMv8 (64-bit)
Cores	1					4	1	4
Clock	700Mhz	700Mhz	700Mhz	700Mhz	700Mhz	900MHz	1GHz	1.2GHz
USB 2.0	2	1		4	1	4	1	4
Ethernet	10/100Mbps	None	10/100Mbps		None	10/100Mbps	None	10/100Mbps
Wi-Fi 802.11n	No							Yes
Bluetooth 4.1	No							Yes
On board storage	SD, MMC, SDIO card slot		4 GB eMMC flash	MicroSDHC slot				
GPIO	8		46	17			40	17
HAT ID bus support	No			Yes			No	yes
GPU	Broadcom VideoCore IV, OpenGL ES 2.0, MPEG-2 and VC-1 , H.264/MPEG-4 AVC							
1080p video	30fps							60fps

Raspberry Pi Pricing

Generation	Release	Model	Price
1	Apr. 2012	Model B	$35
1	Feb. 2013	Model A	$25
n/a	Apr. 2014	Compute Module	$30*
1	Jul. 2014	Model B+	$25
1	Nov. 2014	Model A+	$20
2	Feb. 2015	2 Model B	$35
n/a	Nov. 2015	Zero	$5
3	Feb. 2016	3 Model B	$35

* 100 off price

Global Challenges

Level 1

Climate Change

Water Scarcity

Energy Security

Cyber Security

Global financial structure

Biodiversity and Ecosystem losses

Fisheries Depletion

Deforestation

Infectious Disease

Eliminate our way of life

Level 2

Poverty

Education

The Digital Divide

Urbanization

Intellectual property

International labor and migration

E-Commerce rules

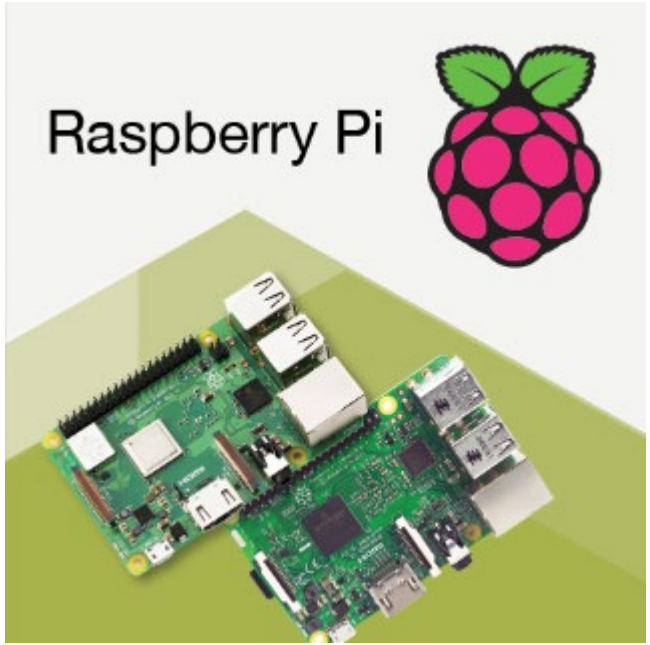
Biotechnology rules

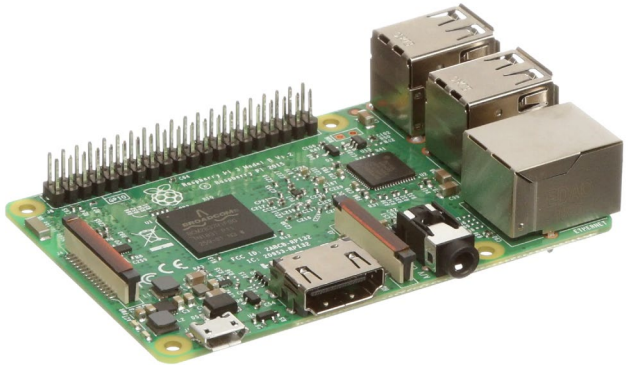
Maritime Safety and Pollution

Disruptive to our way of life

Questions

- Introductions
- What do you know about programming and Raspberry Pi?





Global Competition: SINGAPORE

http://www.country-data.com/cgi-bin/query/r-11828.html

Global Competition: SINGAPORE Today

http://inhabitat.com/marina-bay-sail-by-nbbj-architects/

Accelerated Change!

Years to 50 Million users

Radio – 38 years

Television – 13 years

Cell phone – 7 years

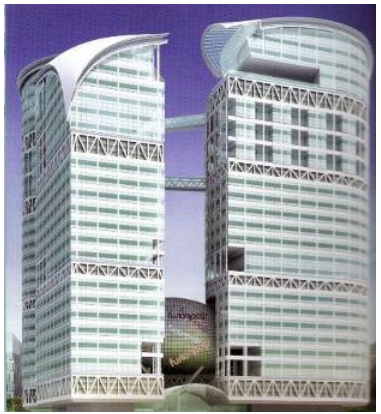
Internet – 4 years

IPOD – 3 years

Facebook – 2 years

More than half of the top 10 in demand jobs in 2014 did not exist in 2004

Shanghai - Pudong

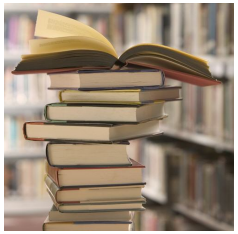


Singapore Science Park

A world of constant disruption…

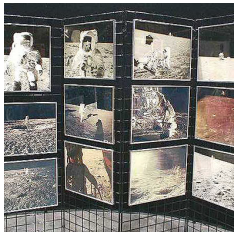
It took **two centuries** to fill the US Library of Congress with more than:

29 million
Books and periodicals



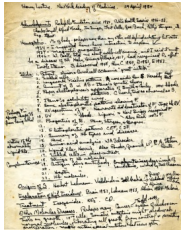
2.4 million
Recordings

12 million
Photographs



4.8 million
Maps

57 million
Manuscripts



Today, it takes about 5 minutes for the world to churn the equivalent amount of new digital information

Grand Opportunities in STEM

In the next 5 years you will no longer need

IDs

Money

Credit Cards

Store cards

Business Cards

Photos

Mail/Mailman

Paper and Hardback Books

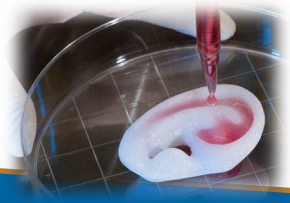
Bills and notices

Paper

Steering Wheels

Organ Donors?

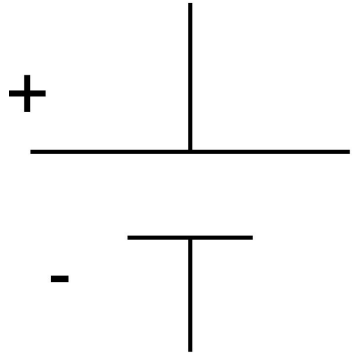
Classrooms?



Introduction to Simple Circuits

Reading Schematics:

What's this?



- Battery

How do electrons start moving in the first place? Well, the force that drives electrons to move, creating current, is called voltage (measured in volts, V). Usually the black terminal is associated with the negative

What is this?

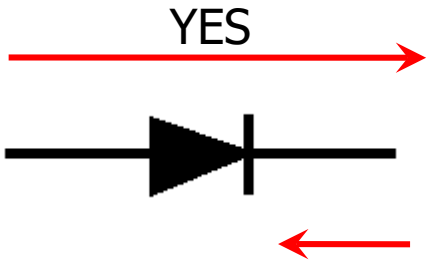
Signal Diode

Power Diode

Zener Diode

Light Emitting Diode - LED

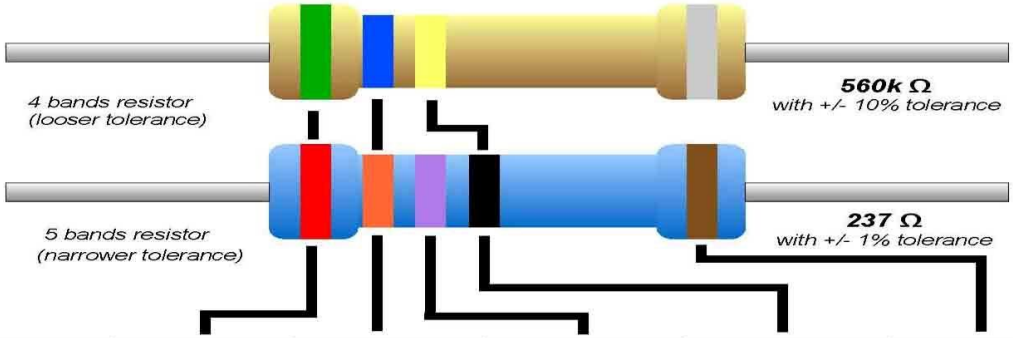
- Diode
  - Lets current flow one direction but not the other



Anode - current flows in (long leg)
Cathode – current flows out

LED

Resistor Color Code

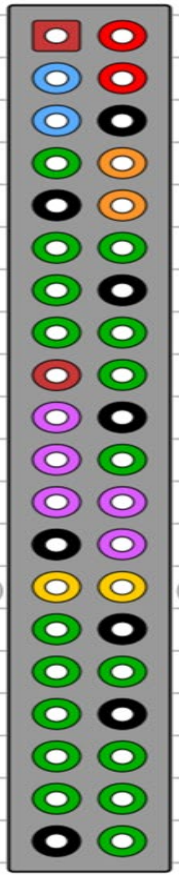


Color	1st Band	2nd Band	3rd Band	Multiplier	Tolerance
Black	0	0	0	x 1 Ω	
Brown	1	1	1	x 10 Ω	+/- 1%
Red	2	2	2	x 100 Ω	+/- 2%
Orange	3	3	3	x 1K Ω	
Yellow	4	4	4	x 10K Ω	
Green	5	5	5	x 100K Ω	+/- .5%
Blue	6	6	6	x 1M Ω	+/- .25%
Violet	7	7	7	x 10M Ω	+/- .1%
Grey	8	8	8		+/- .05%
White	9	9	9		
Gold				x .1 Ω	+/- 5%
Silver				x .01 Ω	+/- 10%

1st band = Orange 3
2nd band = Blue 6 = 360 Ohm
Resistor
Multiplier band = Brown x10 Ohms

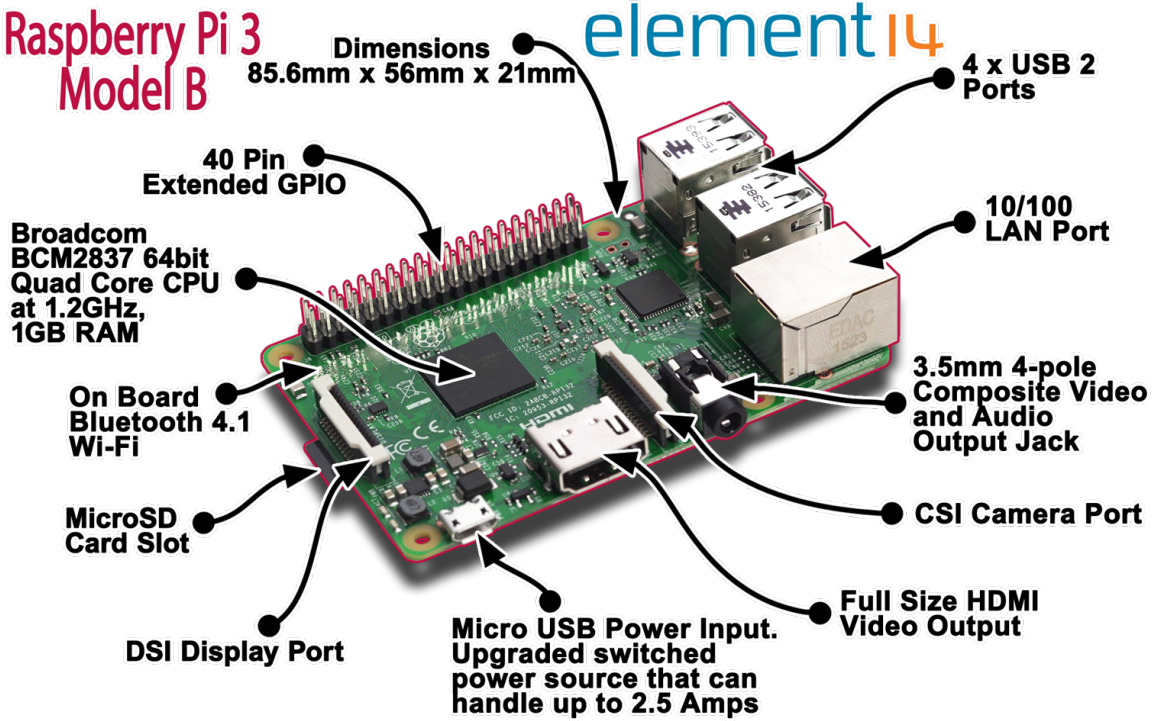
Raspberry Pi 3 GPIO Header

Pin#	NAME		NAME	Pin#
01	3.3v DC Power		DC Power 5v	02
03	GPIO02 (SDA1 , I²C)		DC Power 5v	04
05	GPIO03 (SCL1 , I²C)		Ground	06
07	GPIO04 (GPIO_GCLK)		(TXD0) GPIO14	08
09	Ground		(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)		(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)		Ground	14
15	GPIO22 (GPIO_GEN3)		(GPIO_GEN4) GPIO23	16
17	3.3v DC Power		(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)		Ground	20
21	GPIO09 (SPI_MISO)		(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)		(SPI_CE0_N) GPIO08	24
25	Ground		(SPI_CE1_N) GPIO07	26
27	ID_SD (I²C ID EEPROM)		(I²C ID EEPROM) ID_SC	28
29	GPIO05		Ground	30
31	GPIO06		GPIO12	32
33	GPIO13		Ground	34
35	GPIO19		GPIO16	36
37	GPIO26		GPIO20	38
39	Ground		GPIO21	40



Rev. 2
29/02/2016

www.element14.com/RaspberryPi



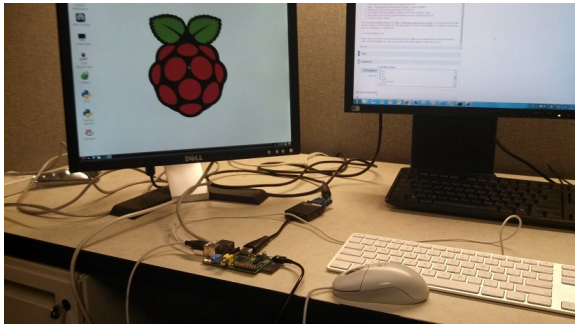
- Connect your Raspberry Pi to your monitor using the HDMI cable. Turn Monitor on.
- Connect your keyboard and mouse
- Connect your power supply

What is Raspberry Pi?

Raspberry Pi is a computer built on a single circuit board with microprocessors, memory, and input/output.

Credit-card sized computer that is used to connect sensors and actuators

It is used as embedded computer controllers.



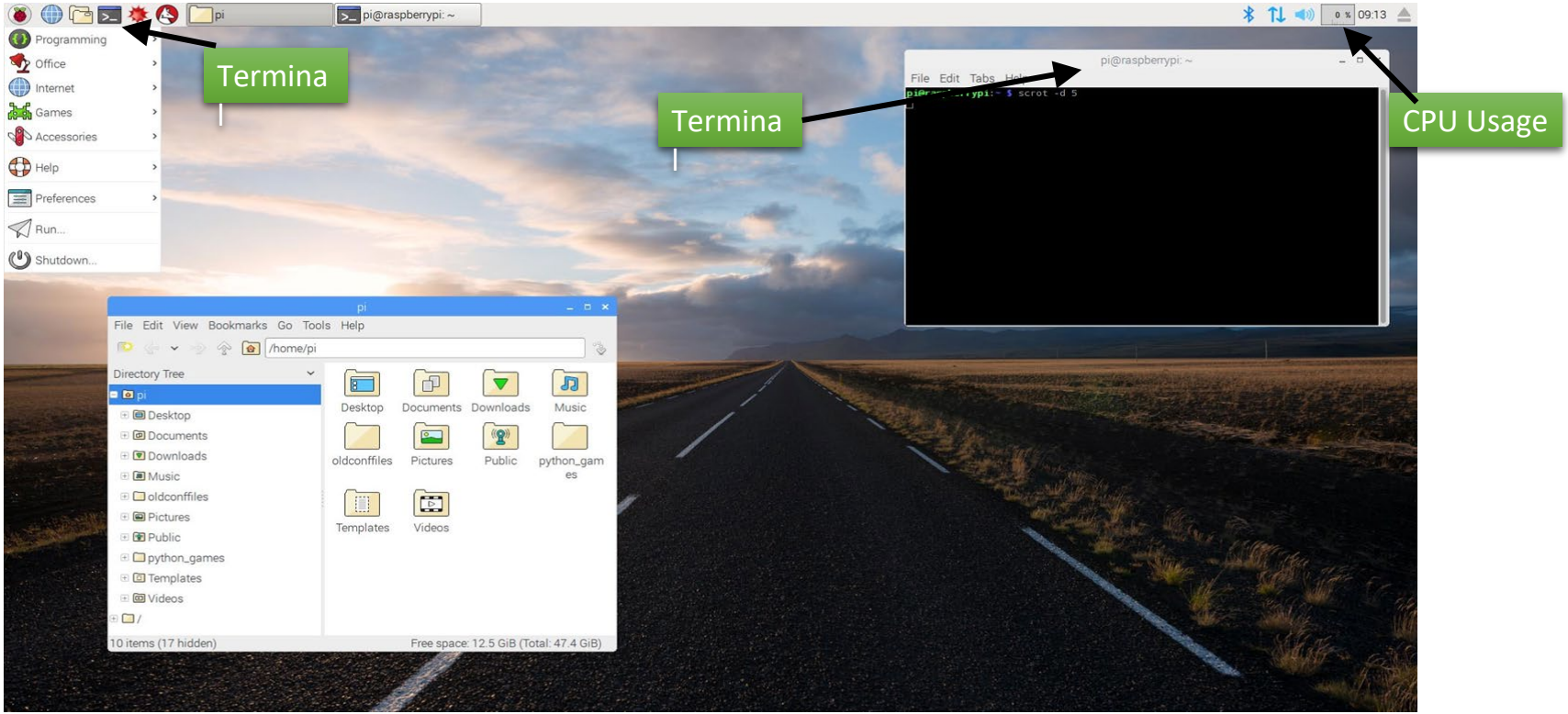
Linux

Open-source Operating system

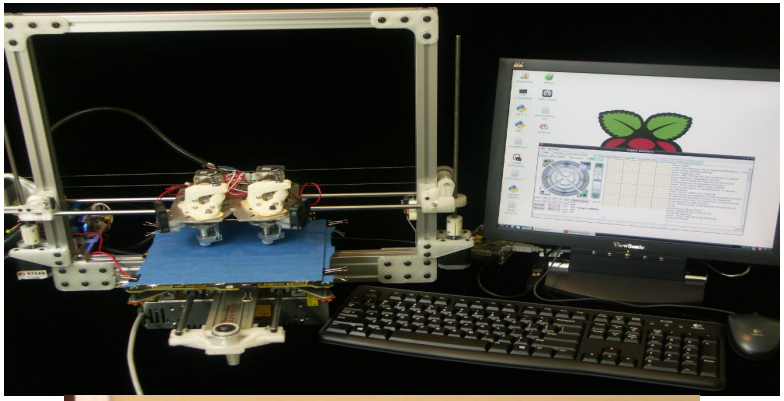
For instance, Windows and MacOS

Raspberry Pi uses Raspbian Jessie,
a Linux-based Operating System

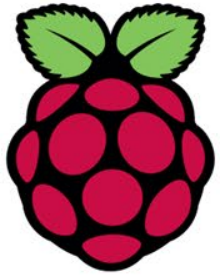
Raspbian Operating System



Some Raspberry-PI Projects



Click on the images to see the videos on YouTube.



Let’s Get Physical

Scratch & Python

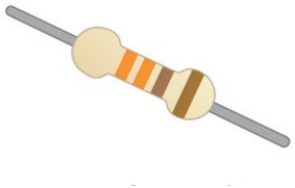
Goals

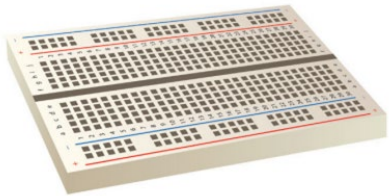
- Why physical computing?
- Build a simple circuit using Raspberry Pi
- Control an output using Scratch
- Create more complex circuits & programs with Python

Workshop outline

You will need:

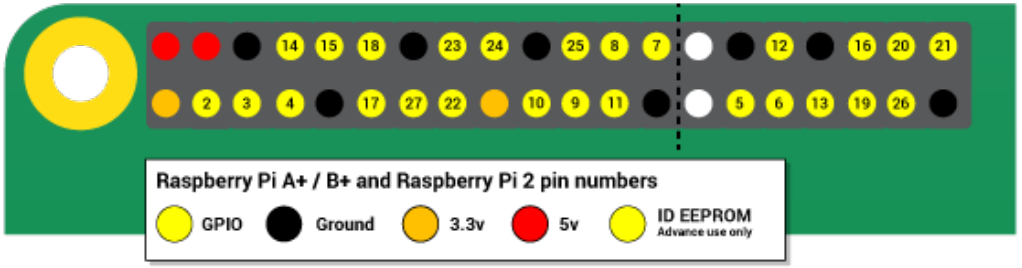
x 2

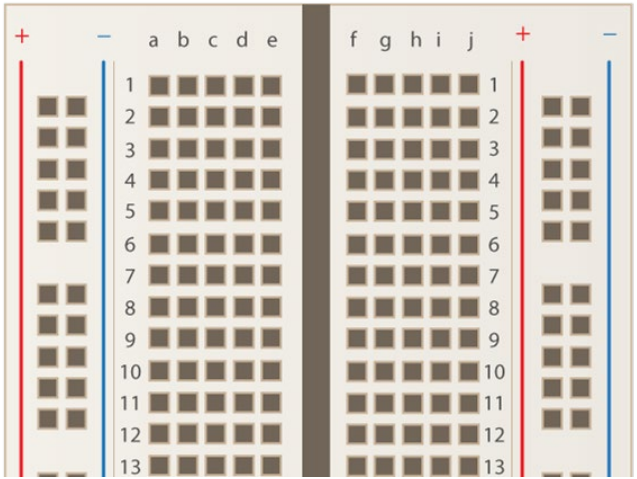


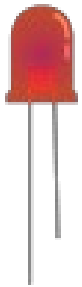


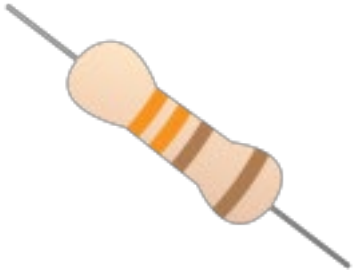
Physical Computing

Enables creators to connect the virtual world with the physical, great for engagement.

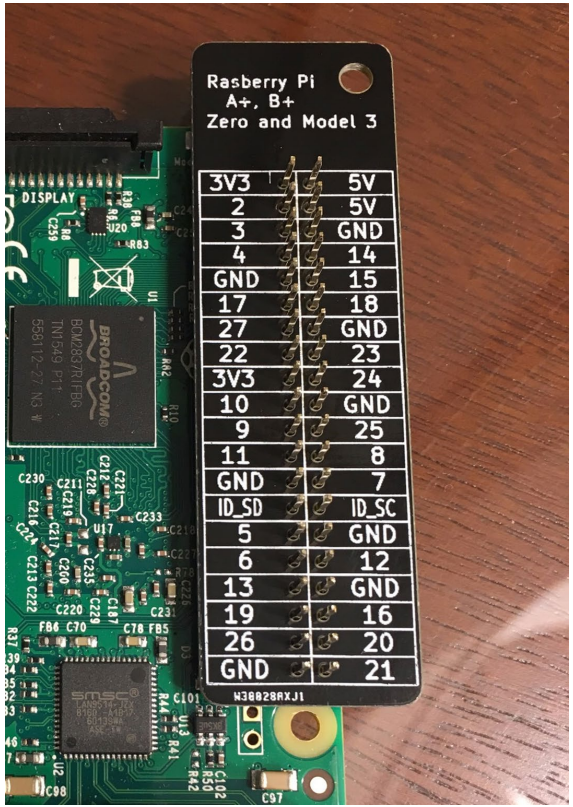


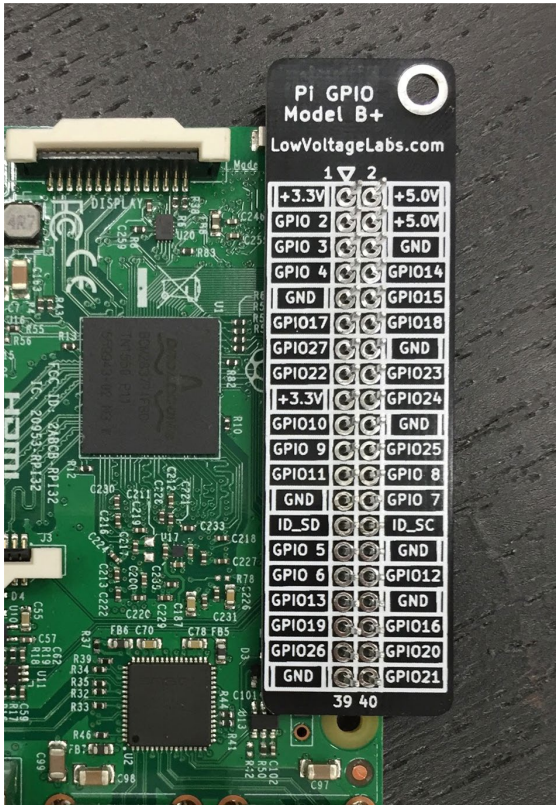






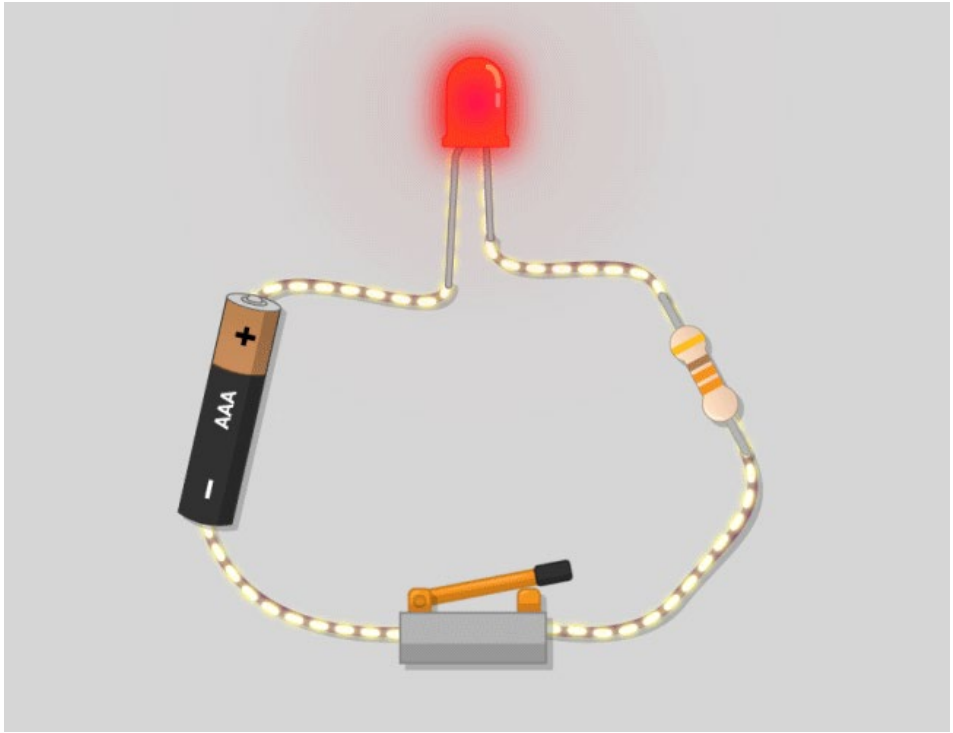
GPIO reference board



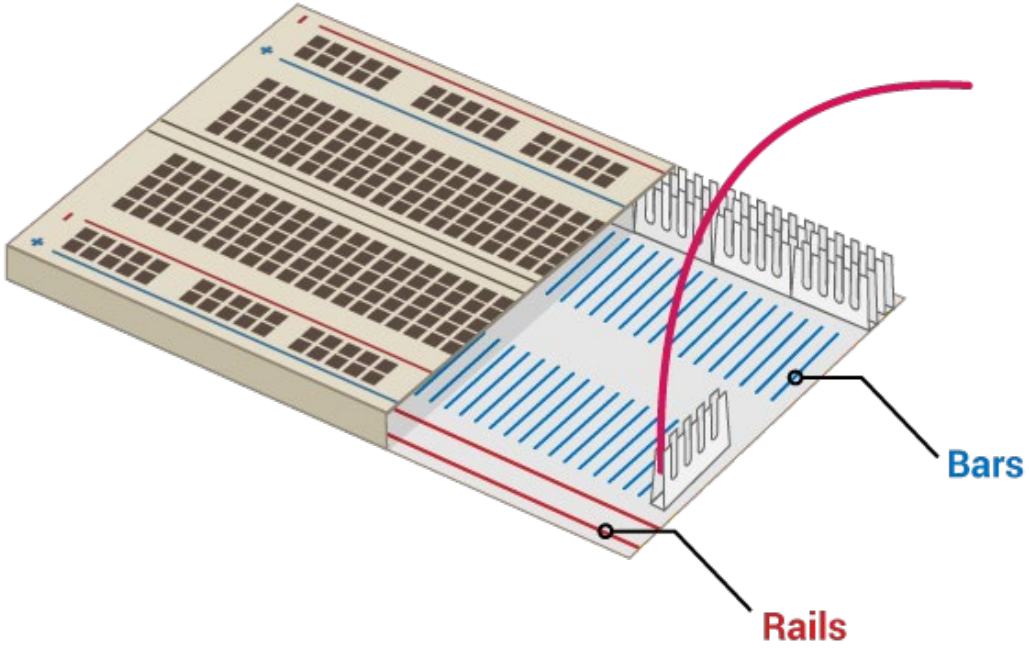


Simple Circuit

Your Raspberry Pi can act as the power supply for simple circuits, we can use this test our hardware is working.

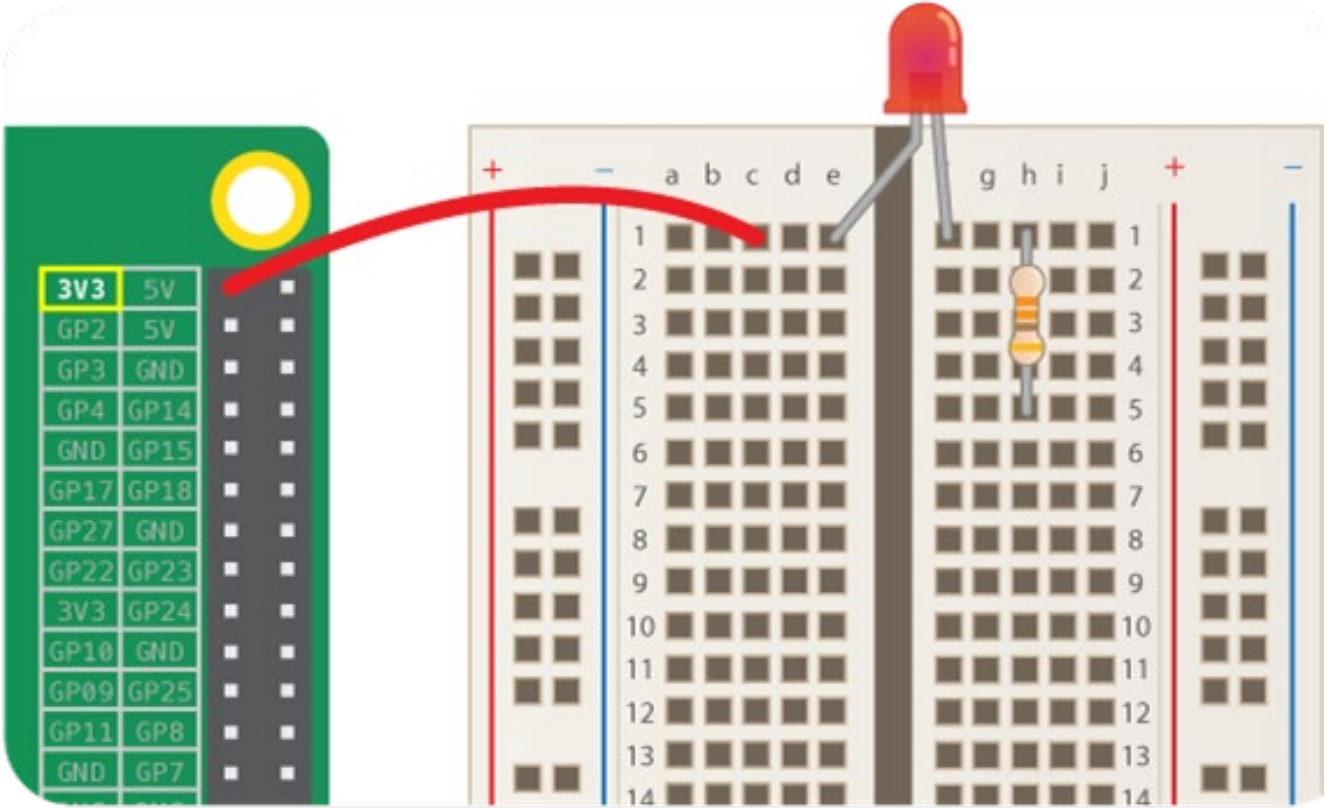


Inside a breadboard



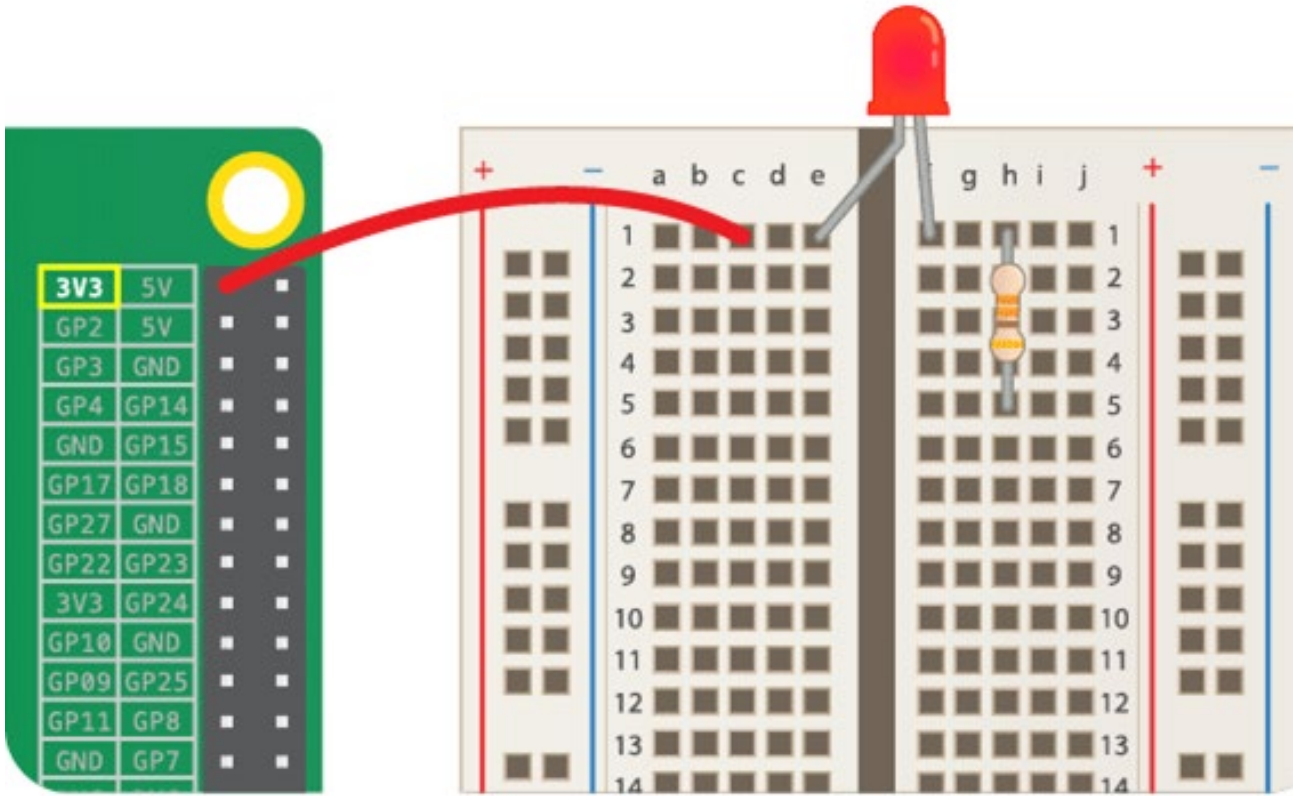
Simple Circuit

Your Raspberry Pi can act as the power supply for simple circuits, we can use this test our hardware is working.



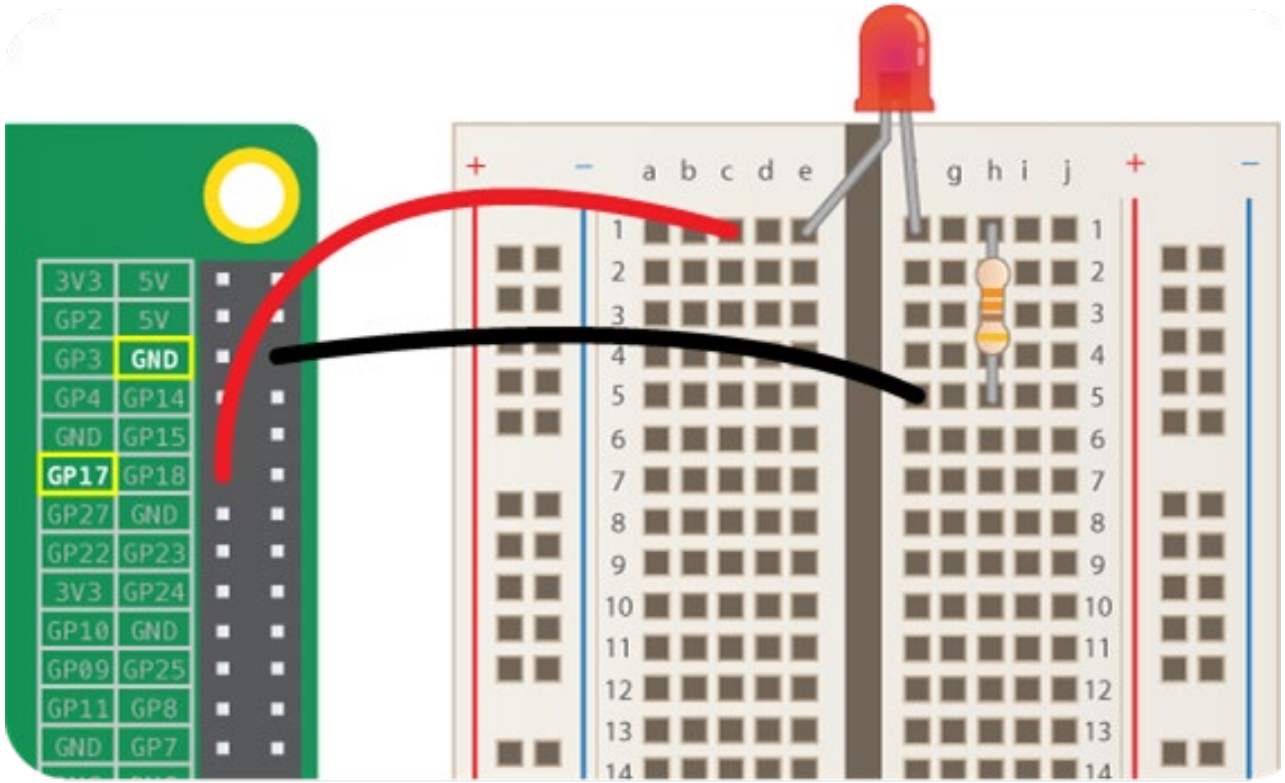
Simple Circuit

Your Raspberry Pi can act as the power supply for simple circuits, we can use this test our hardware is working.



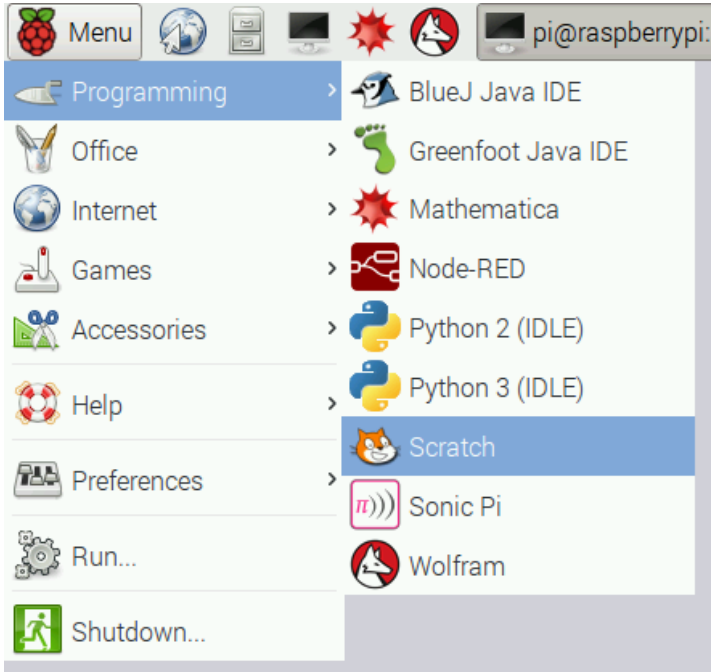
Switching Circuit

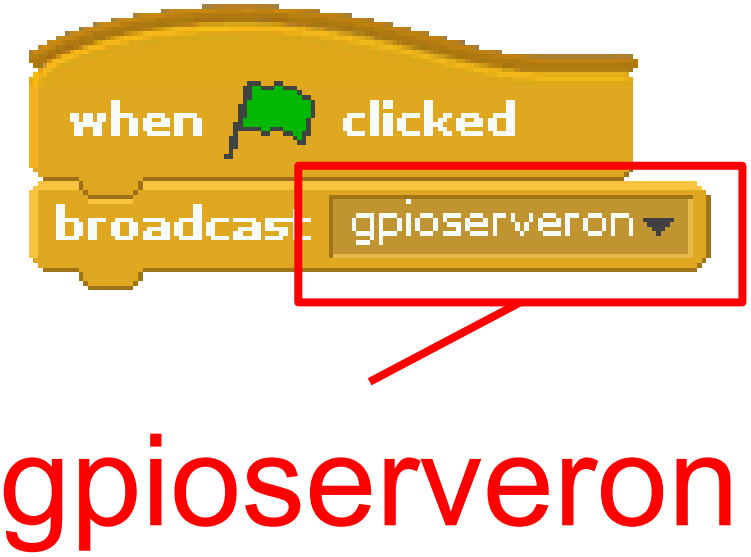
Your Raspberry Pi can act as a switch in the circuit, which can be controlled in software. Connect the positive leg of the LED to any other GPIO pin on your Raspberry Pi.



Scratch on the Raspberry Pi

Open Scratch and activate the GPIO server.





Blinking LED

Setup your pin as an output.		config17out
Switch pin 17 on		gpio17on
Pause		
Switch pin 17 off		gpio17off
Pause		
Loop forever		

Extra Challenge!

Can you:

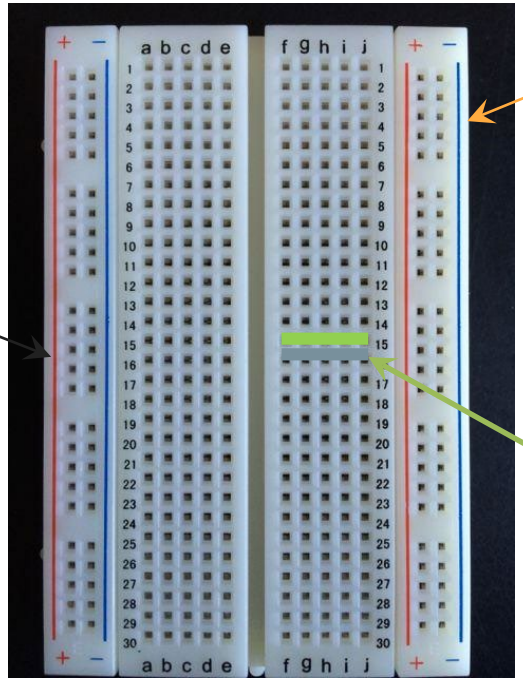
- Flash your LED at different speeds, how fast can you make it flash?
- Can you make a dot (short flash) and dash (long flash) and use to make a distress beacon. S(...) O(---) S(...)

Programming in Python

- Python is a high-level programming language. By high level we mean that it is more removed from the actual language and details of a computer and utilizes more natural language in its operation.
- This high-level language makes it easier to use
- We will learn some basic Python in this class. For those wanting to know more I recommend going through the online tutorial:

https://www.codecademy.com/learn/python

The Breadboard



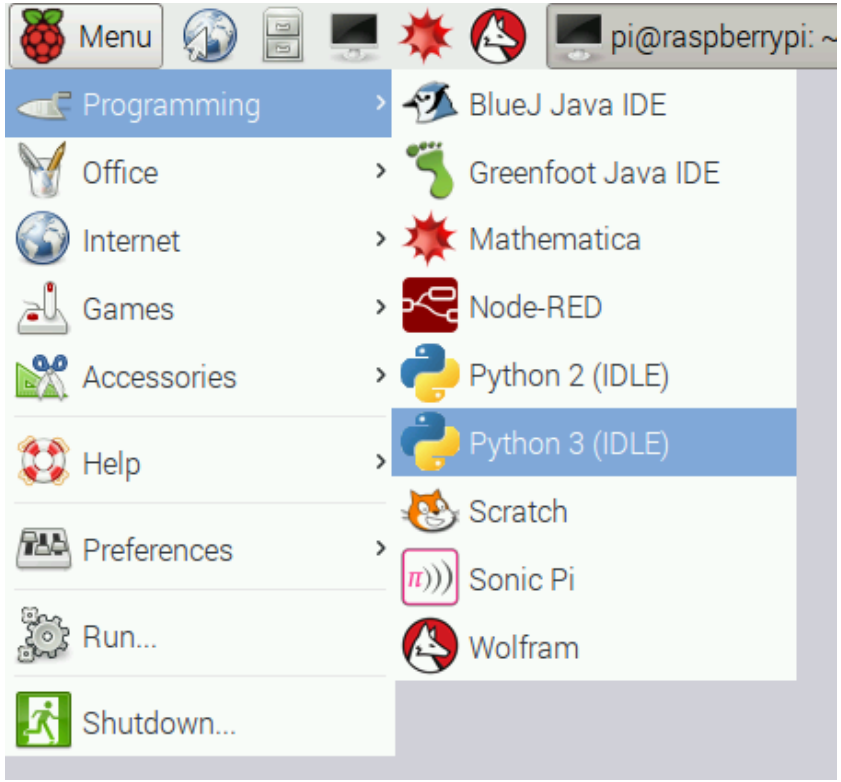
Every hole along the red line (this column) is electrically connected vertically, but not horizontally

Every hole along the blue line (this column) is electrically connected vertically, but not horizontally

Every hole along a given row (green line) is electrically connected horizontally, but NOT vertically

The two red and two blue columns are called buses

Create New Python 3 File



Coding Your LED Light

```
# Flashing LED


from gpiozero import LED
from time import sleep

myled = LED(17)

while True:
    myled.on()
    sleep(1)
    myled.off()
    sleep(1)
```


Example Programs

```
# Flashing LED

from gpiozero import LED

led = LED(17)

led.blink()
```

Try some other values in led.blink(), what would these do:

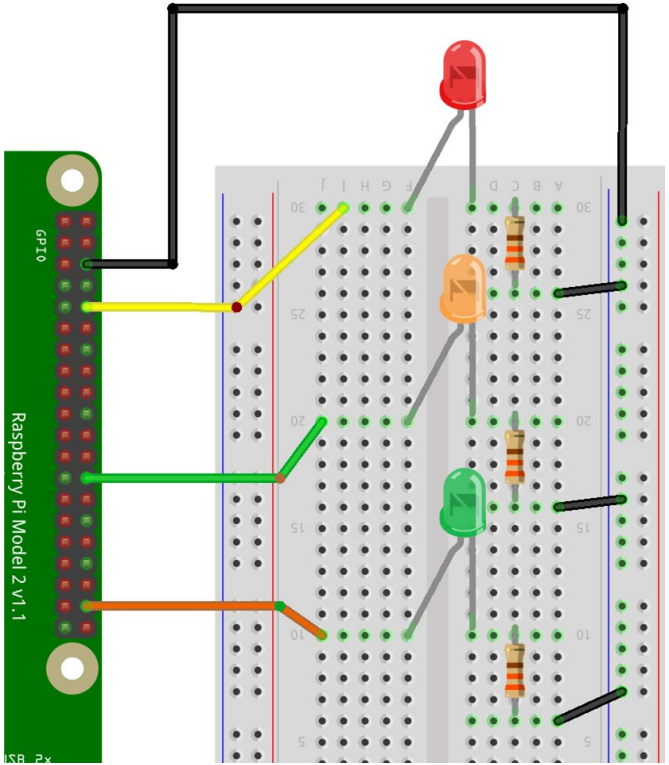
- led.blink(5)
- led.blink(2, 0.5)
- led.blink(0.1, 10)
- led.blink(0.5, 0.5, 5, False)

Example Programs

```
# LED methods from the docs:
https://gpiozero.readthedocs.io

led.on() - Switches the Pin high

led.off() - Switches the Pin low

led.blink() - Makes the LED blink

led.toggle() - Change the state of the LED

led.pin.number - Returns the pin number

led.is_lit - Returns the current state
```


Abstraction

```
# Traffic Lights 1

from gpiozero import LED
from time import sleep

red = LED(21)
amber = LED(20)
green = LED(16)

red.on()
sleep(3)
red.off()
amber.on()
…
```



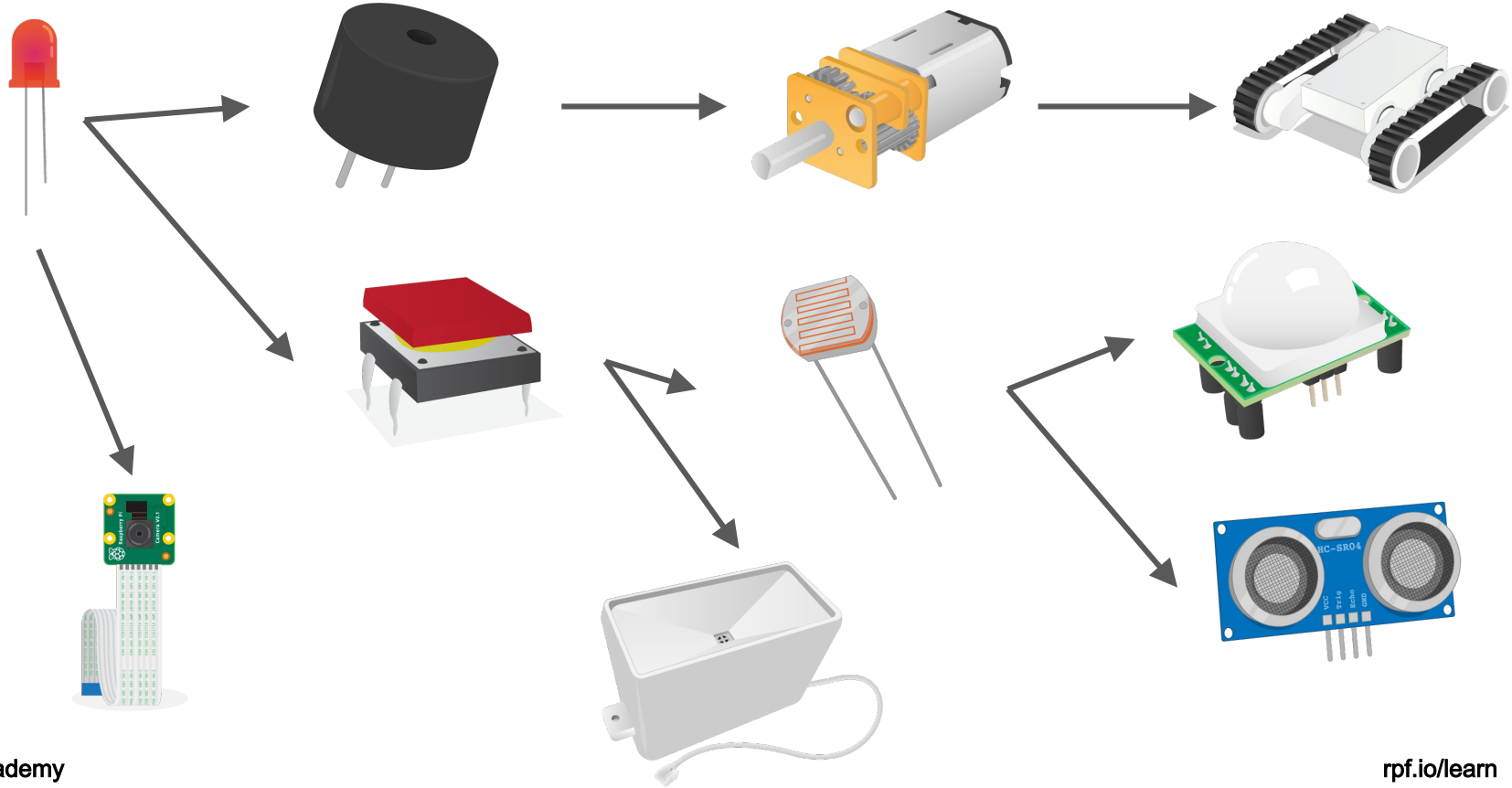
Abstraction

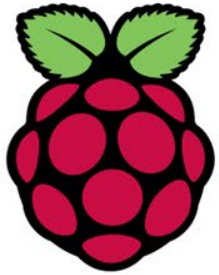
Removing complexity to make a task more accessible

```
# Traffic Lights 1

from gpiozero import LED
from time import sleep

red = LED(21)
amber = LED(20)
green = LED(16)

red.on()
sleep(3)
red.off()
amber.on()
…
```

```
# Traffic Lights 2

from gpiozero import TrafficLights
from time import sleep

lights = TrafficLights(21, 20, 16)



lights.red.on()
sleep(3)
lights.red.off()
lights.amber.on()
…
```


Where next?



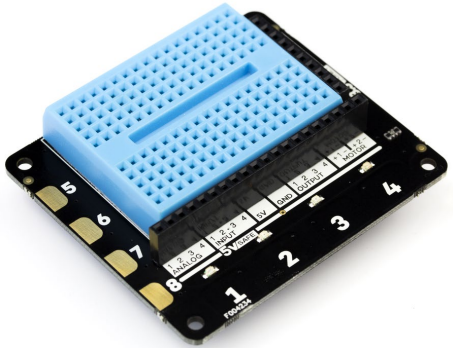


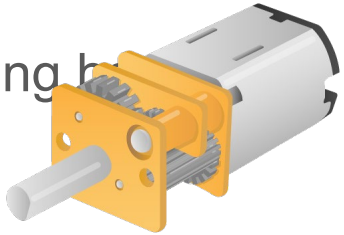
Make things SPIN!

Explorer Hat & Python

What you will learn

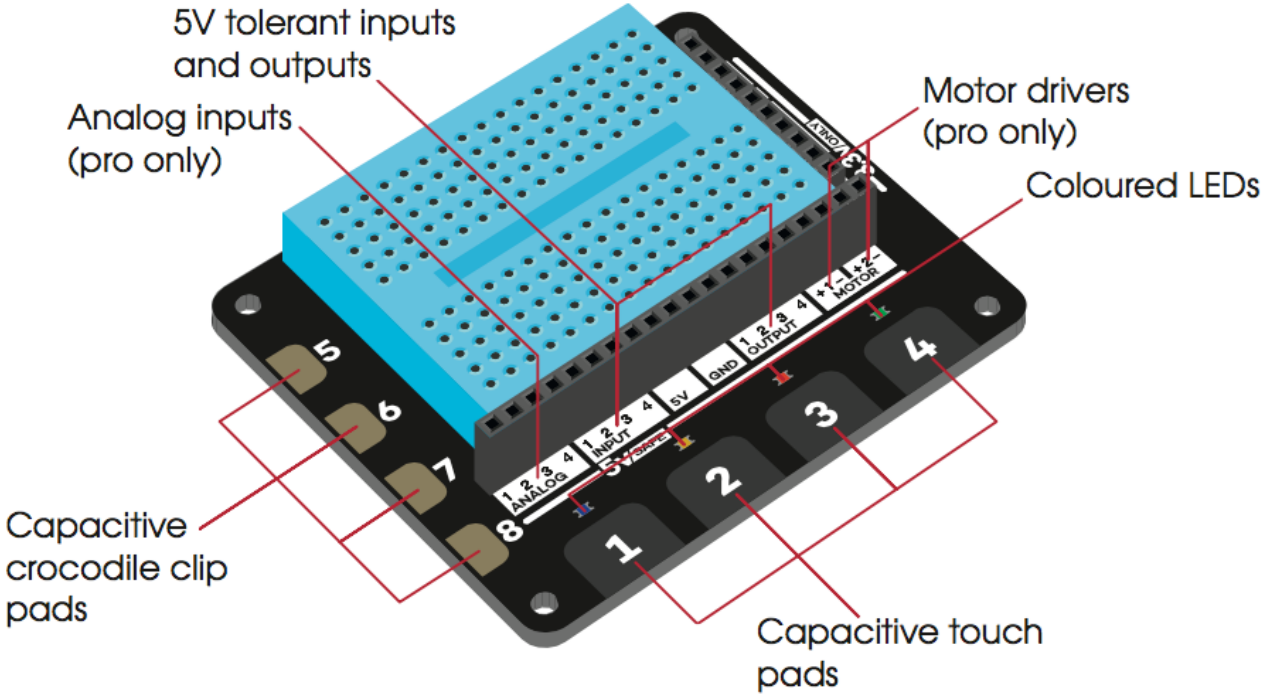
- What is a HAT?
- To turn on and off LEDs with Python
- How to control motors
- To use capacitive touch buttons to make something h
- Bring it all together and create an invention





Explorer HAT Pro by Pimoroni

Add-on board for physical computing with Raspberry Pi.



Putting your HAT on

First power down and unplug the power cable. Then:

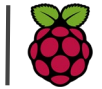
Testing the Explorer HAT in Python

```
>>> import explorerhat
```

It should display a message
“Explorer HAT Pro detected…”

Testing the Explorer HAT in Python

```
>>> explorerhat.light.red.on()
>>> explorerhat.light.red.off()

>>> explorerhat.light.red.toggle()
>>> explorerhat.light.red.blink(0.5, 0.2)

>>> explorerhat.light.on()
>>> explorerhat.light.off()

>>> explorerhat.light.blue.pulse(0.2, 0.2, 0.5, 0.5)
```

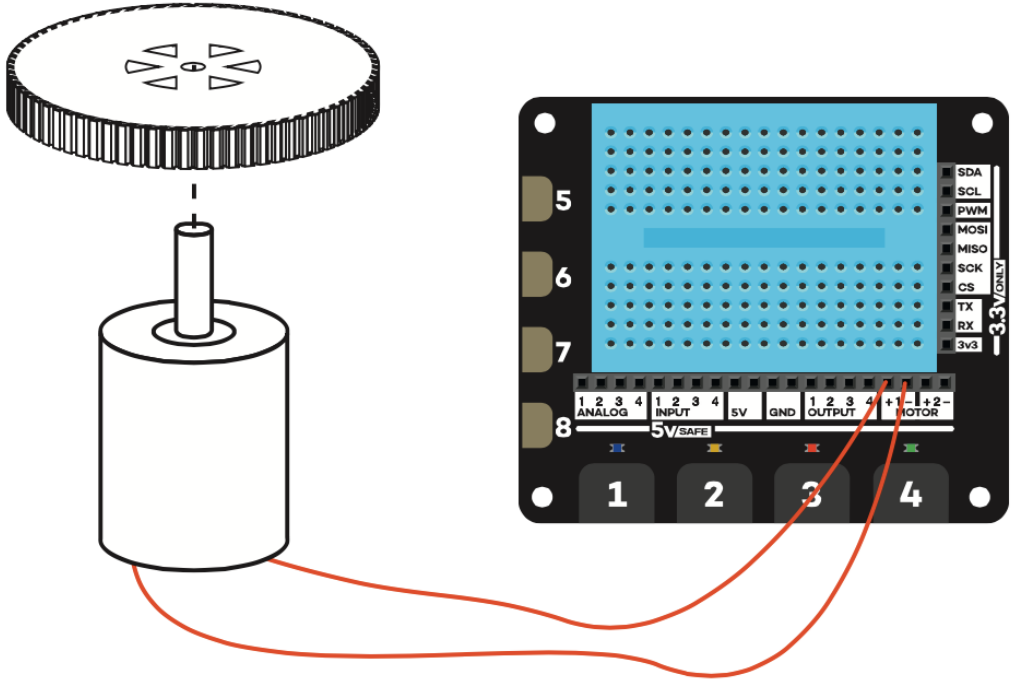
Challenges

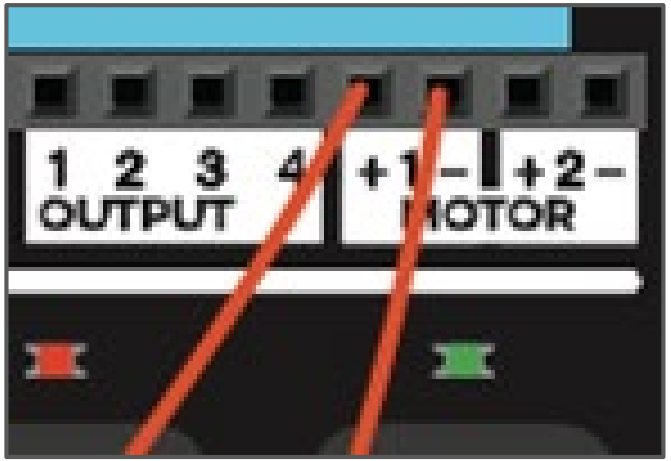
Play around with the other colours “yellow”, “green”, “blue”:

- Can you light these LEDs up?
- Open a **New Window**, via the **File** menu and create a light sequence.
- What activities could your students do with this?

Connecting a motor

The Explorer HAT Pro has some circuitry to make controlling motors easier.





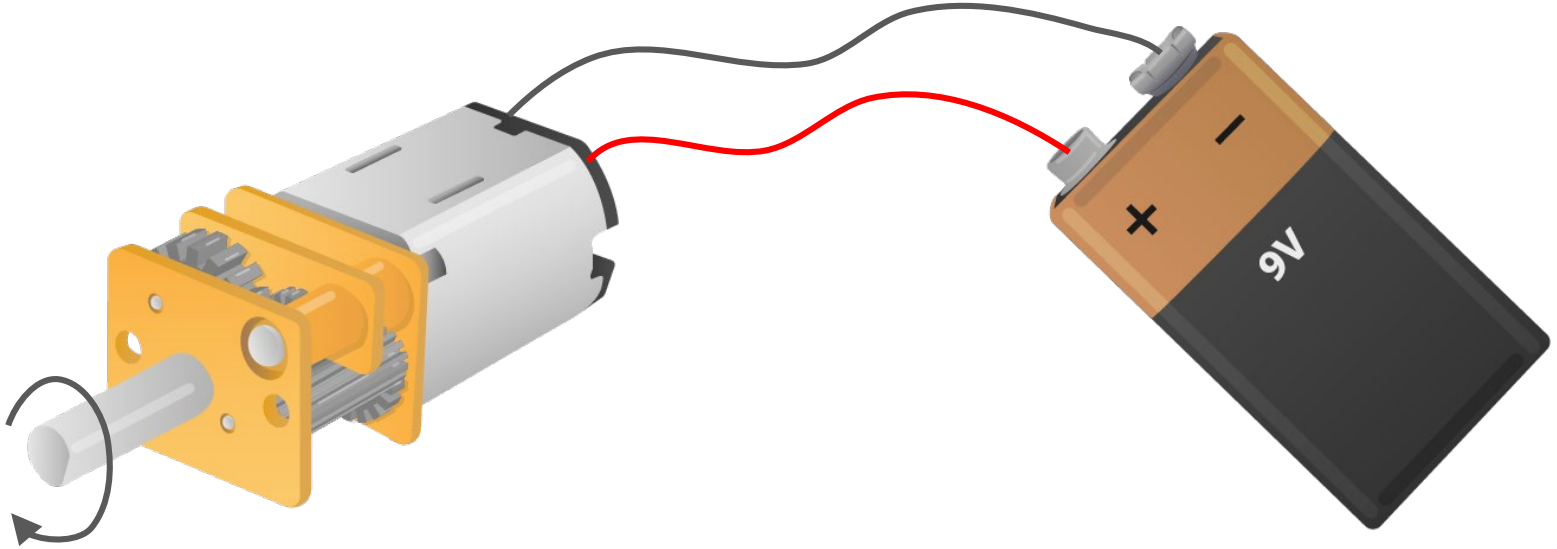
Code it

In a **new Python 3 file** window type:

```
import explorerhat
from time import sleep

explorerhat.motor.one.forward(100)

sleep(5)

explorerhat.motor.one.stop()
```


Why do we need the Explorer HAT for this?

The HAT has an H-Bridge. Here's what it's for:



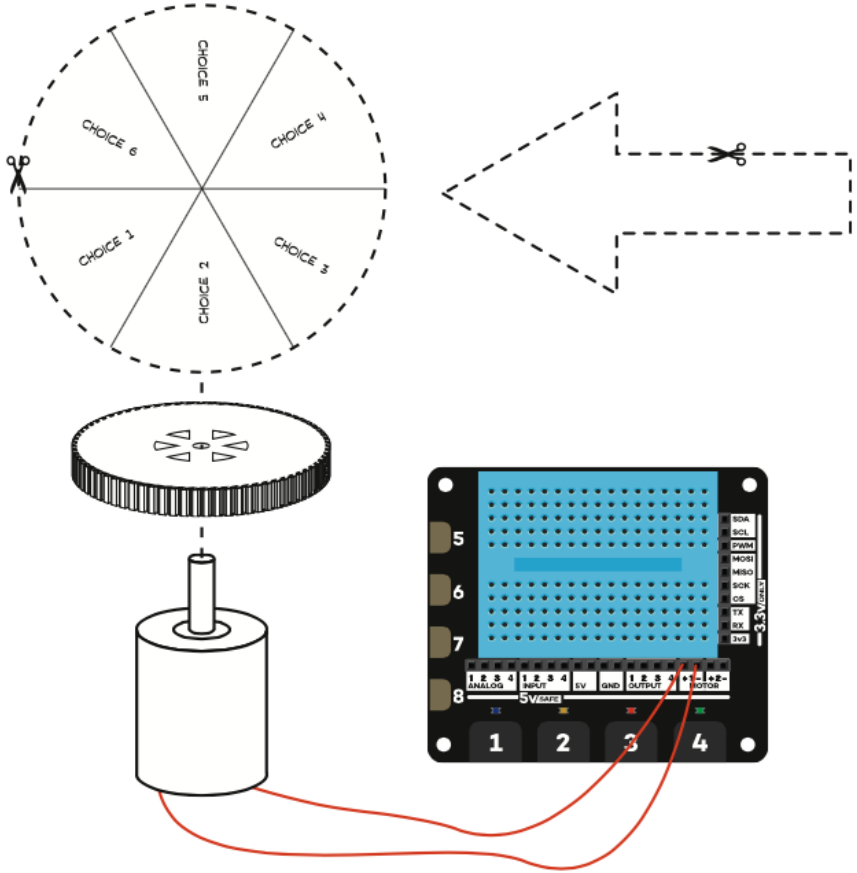
Touch input and Motor Control

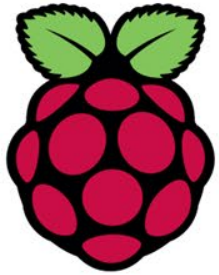
```
import explorerhat
from time import sleep

def wheel (channel, event):
    explorerhat.motor.one.forward(100)
    sleep(5)
    explorerhat.motor.one.stop()

explorerhat.touch.one.pressed(wheel)
```

Touch input and Random Motor Control

```
import explorerhat
from time import sleep

from random import randint

def wheel (channel, event):
    duration = randint(1, 10)
    print(duration)
    explorerhat.motor.one.forward(100)
    sleep(duration)
    explorerhat.motor.one.stop()

explorerhat.touch.one.pressed(wheel)
```


Make something!



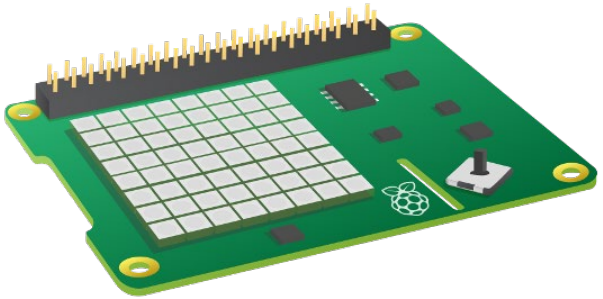


Sense HAT

...and sense-ability!

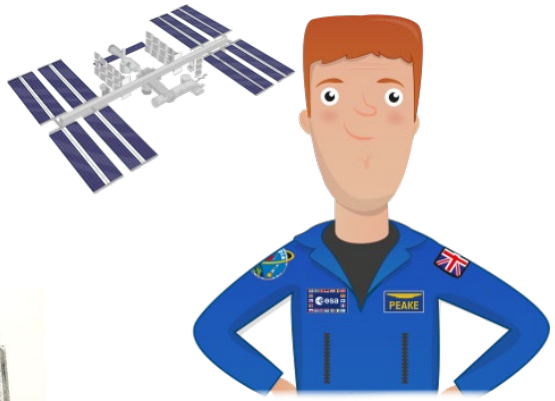
What you will learn

- Hardware, Emulator, and Web
- Displaying emojis using RGB LEDs
- Sensing the orientation of the Sense HAT
- Using Python to sense the environment

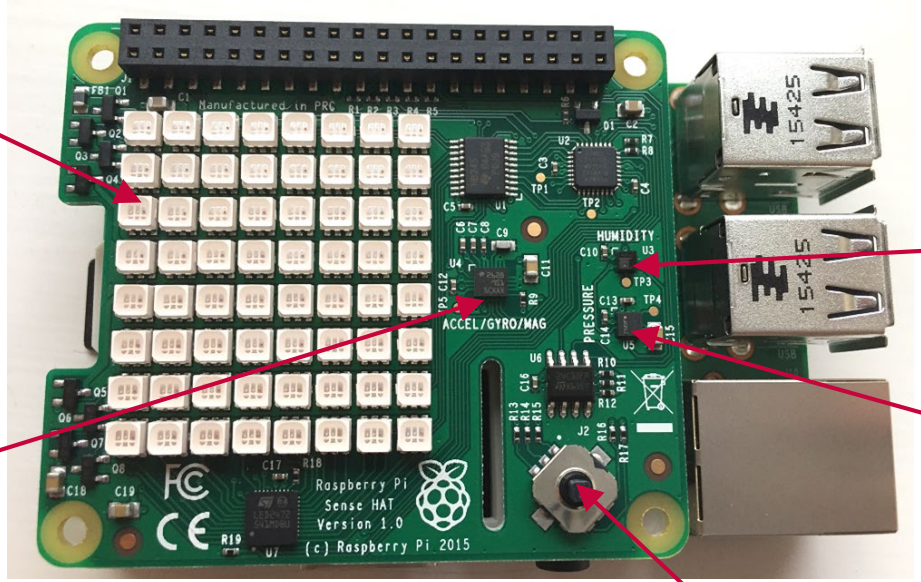


Features of the Sense HAT

The Sense HAT was designed for the Astro Pi project



8x8 RGB LED matrix



Temperature and humidity sensor

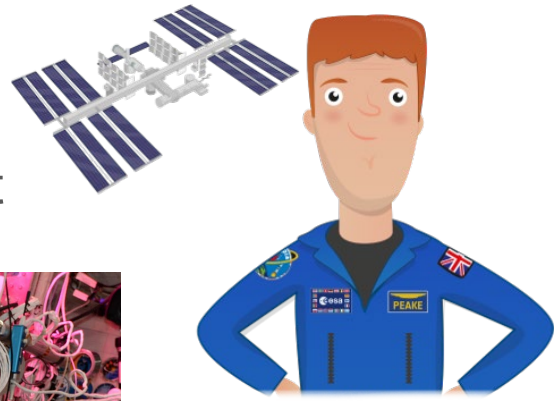
Barometric pressure sensor

Gyroscope, accelerometer and magnetometer

Mini joystick

Astro PI

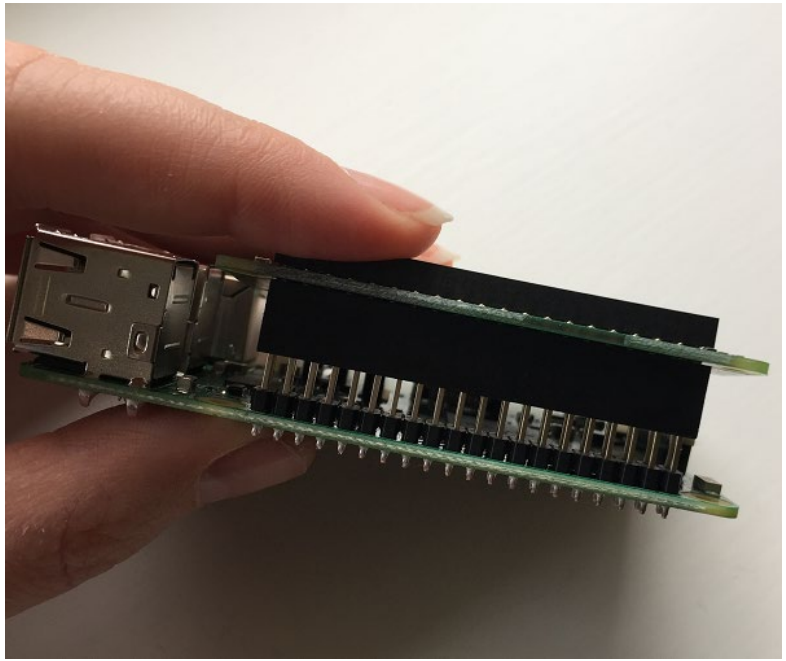
The Sense HAT was designed for the Astro Pi project

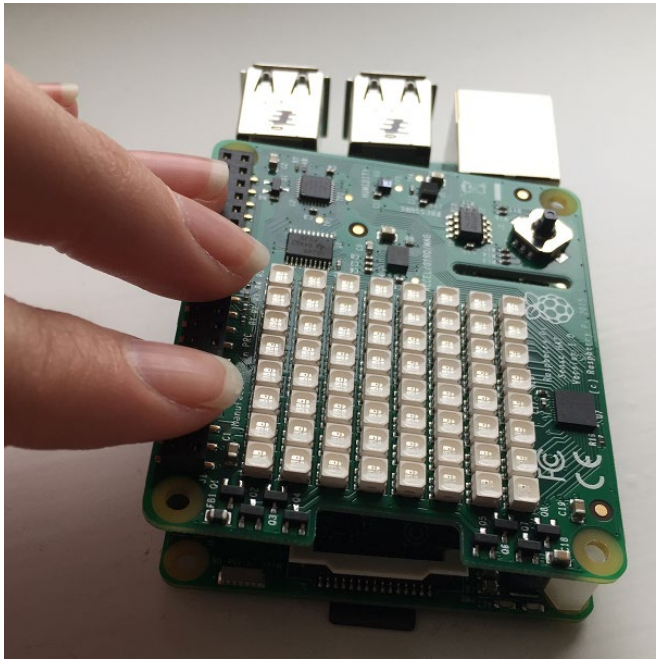


Attaching your Sense HAT

Power down and unplug your Raspberry Pi

Align the pins with the Sense HAT and push gently



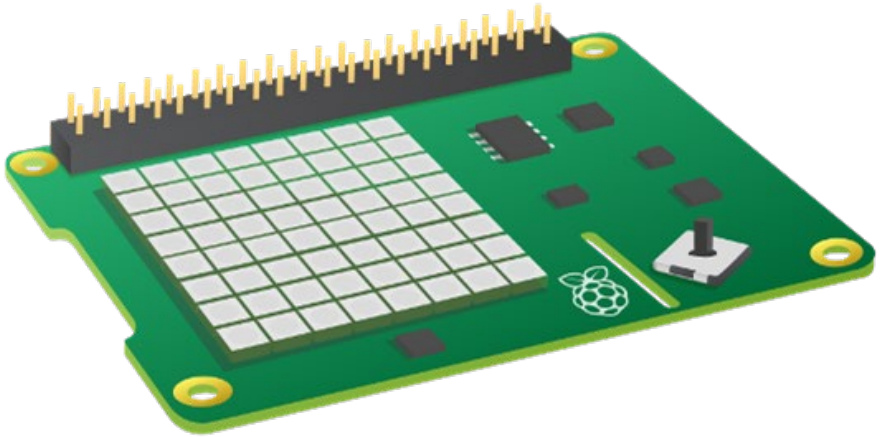


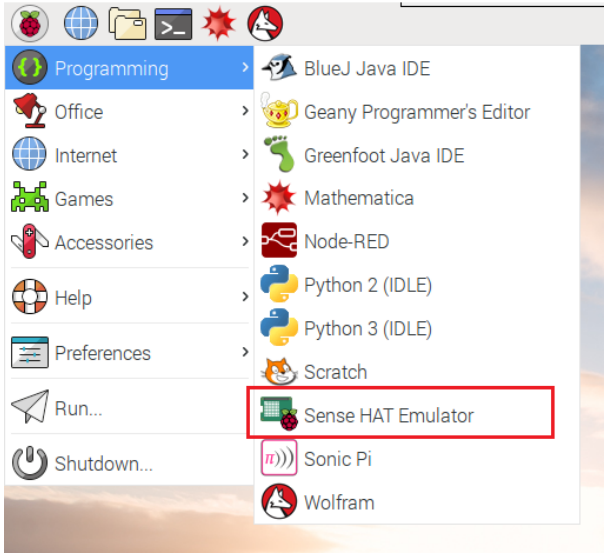
Sense HAT

Add-on board for physical computing with Raspberry Pi

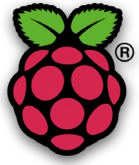
Software emulator on the Raspberry Pi

Web-based emulator - https://trinket.io/sense_-hat





Programming the Sense HAT with Python



Programming → Python 3 (IDLE)

```
from sense_hat import SenseHat

sense = SenseHat()

sense.show_message("Picademy")
```

While I love how the emulator allows for different experimentation and doesn't require a board purchase, I fall for those shiny LEDs every time.

Displaying colours (US: colors)

In a new Python 3 file window type:

```
from sense_hat import SenseHat
from time import sleep

sense = SenseHat()

red = (255, 0, 0)
sense.clear(red)                    # Fill whole screen red
sleep(1)
sense.clear()                                          # Fill whole screen
blank
sense.set_pixel(4, 5, red)  # Set specific pixel red
```

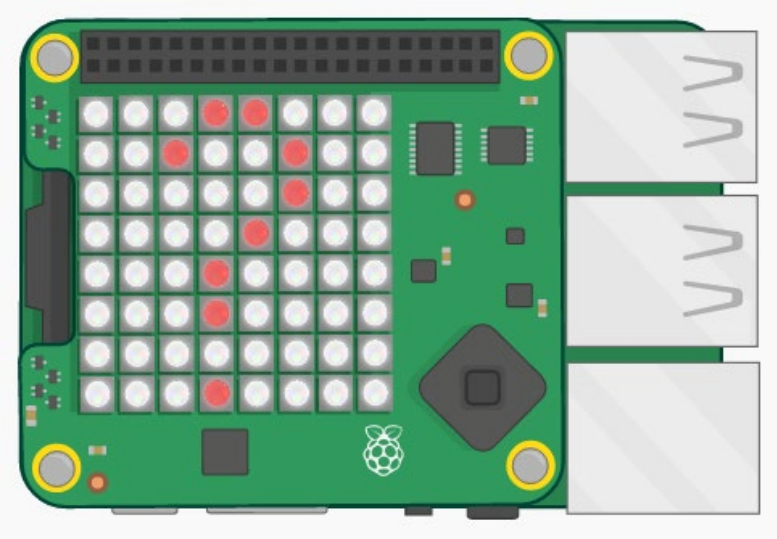
Can you work out how to create green? Or blue? What about magenta or cyan?

You can also use “sense.clear(xxx, xxx, xxx,)” to play with the colors.

Displaying emojis

https://pythonhosted.org/sense_hat/api/ - set_pixels

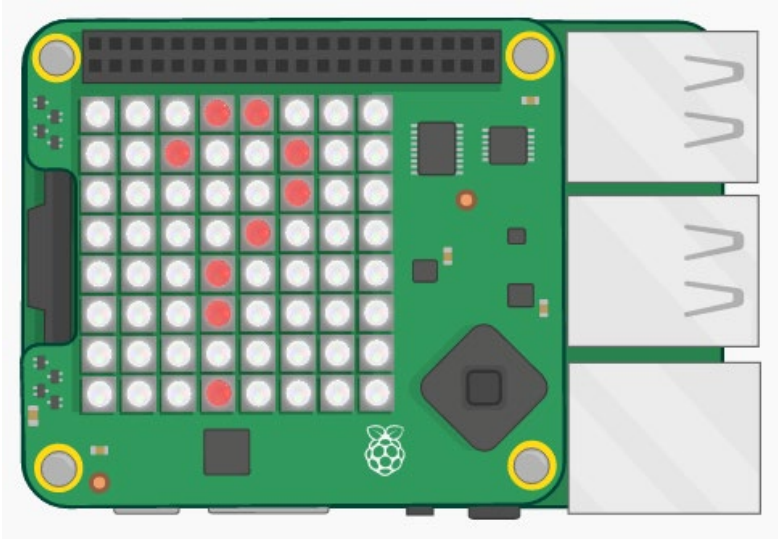
```
X = (255, 0, 0)
          # Red
O = (255, 255, 255)
          # White

question_mark = [
O, O, O, X, X, O, O, O,
O, O, X, O, O, X, O, O,
O, O, O, O, O, X, O, O,
O, O, O, O, X, O, O, O,
O, O, O, X, O, O, O, O,
O, O, O, X, O, O, O, O,
O, O, O, O, O, O, O, O,
O, O, O, X, O, O, O, O
]

sense.set_pixels(question_mark)
```



Getting emoji -onal

Can you write a program that will:

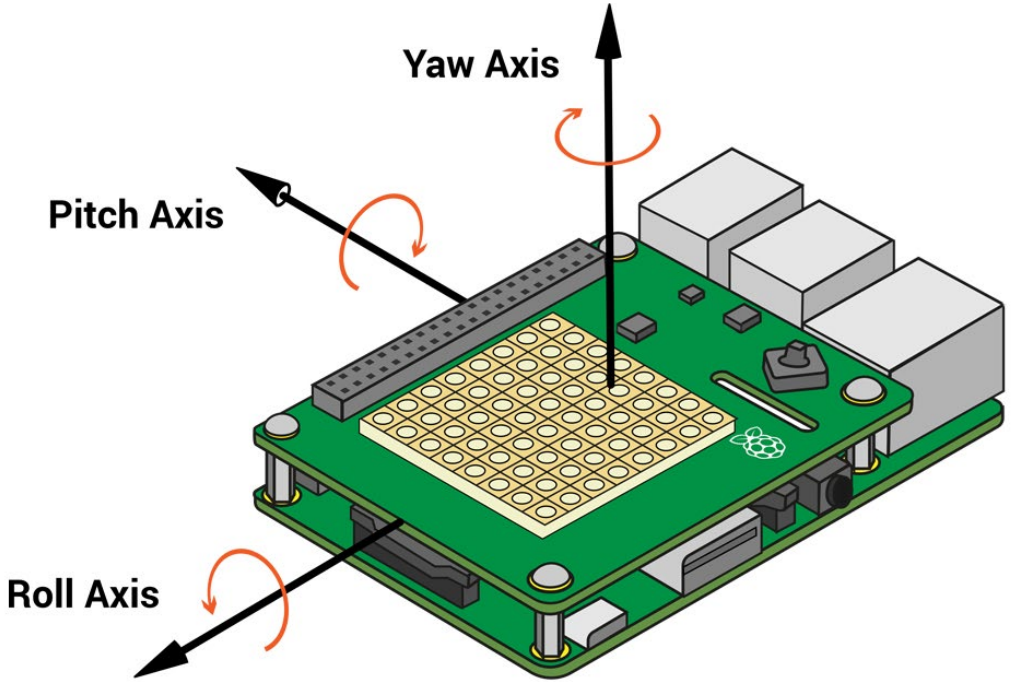
- Display a different emoji
- ...depending on either the humidity
- ...or the orientation of the Sense HAT?
- Make a marble maze
- Animate a pixel creature



For the curious:
https://pythonhosted.org/sense-hat/api/

Sensing the orientation of the Sense HAT

The Sense HAT can report **pitch**, **roll** and **yaw**



Sensing the orientation of the Sense HAT

The Sense HAT can report **pitch**, **roll** and **yaw**

```
from sense_hat import SenseHat

sense = SenseHat()

data = sense.get_orientation()
pitch = data['pitch']

print( pitch )
```

Sensing the orientation in ACTION!

The Sense HAT can report **pitch**, **roll** and **yaw**

```
from sense_hat import SenseHat

sense = SenseHat()

while True:
    data = sense.get_orientation()
    pitch = data['pitch']

    print( pitch )
```

What causes the value to change? Replace ‘pitch’ with ‘yaw’ or ‘roll’. Now what causes the value to change?

```
blank = (0,0,0)                                    #don’t forget to import
your
white = (255,255,255)                              #modules
sense.clear()
x = 1
y = 1
while True:
    data = sense.get_orientation()
        pitch = data['pitch']

    if 359 > pitch > 179 and x != 7 :
        sense.set_pixel(x, y, blank)                    # Clear pixel
        x += 1
        # Move x coord
        sense.set_pixel(x, y, white)                    # Set pixel

    elif 1 < pitch < 179 and x != 0:
        # What should you do here?
```

Python - Sensing humidity

```
from sense_hat import SenseHat
from time import sleep

sense = SenseHat()

while True:
    humidity = sense.get_humidity()

    if humidity > 29:          #change value based on room
        sense.clear(255,0,0)
    else:
        sense.clear(0,0,255)

    sleep(0.1)
```


Sensing Pressure and Humidity

The Sense Hat can also display **pressure** and **humidity**

```
from sense_hat import SenseHat
Sense = SenseHat()

While True:
            t = sense.get_temperature()
            p = sense.get_pressure()
            h = sense.get_humidity()

            t = round(t, 1)
            p = round(p, 1)
            h = round(h, 1)

            msg = ‘Temperature = {0}, Pressure = {1}, Humidity = {2}’.format(t, p, h)

            sense.show_message(msg, scroll_speed = 0.05)
```


Common Sense Hat Commands

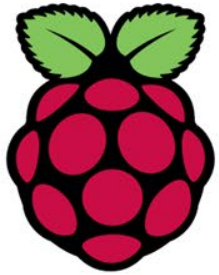
https://pythonhosted.org/sense-hat/api/

*.show_message("text") #Can add conditions

*.clear() #sets all pixels to condition

*.pixel(x, y, *variable*) #sets x, y, pixel to *v*

```
Conditions:
    scroll_speed=x.xx
    text_colour[xxx,xxx,xxx]
    back_colour[xxx,xxx,xxx]
```

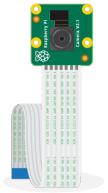



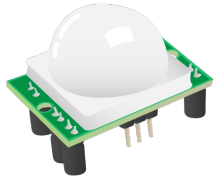
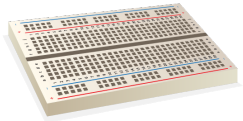
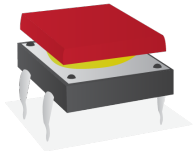
Raspberry Pi Camera and other inputs

What you will learn

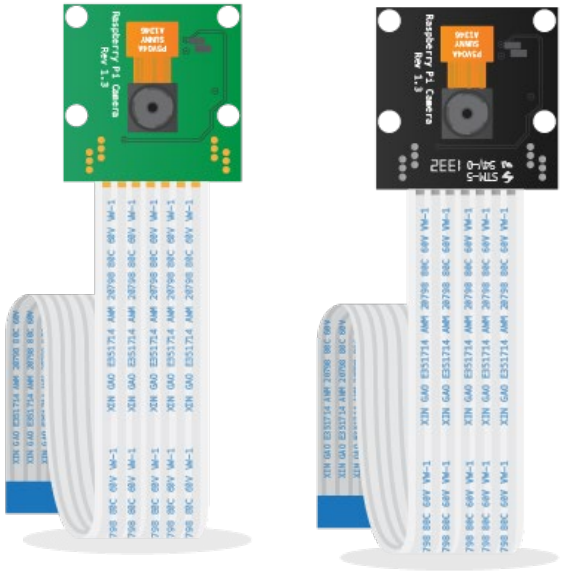
- How to connect the camera module
- How to use Python to take pictures
- How to add physical components to your project
- How to use loops to repeat commands

You will need:



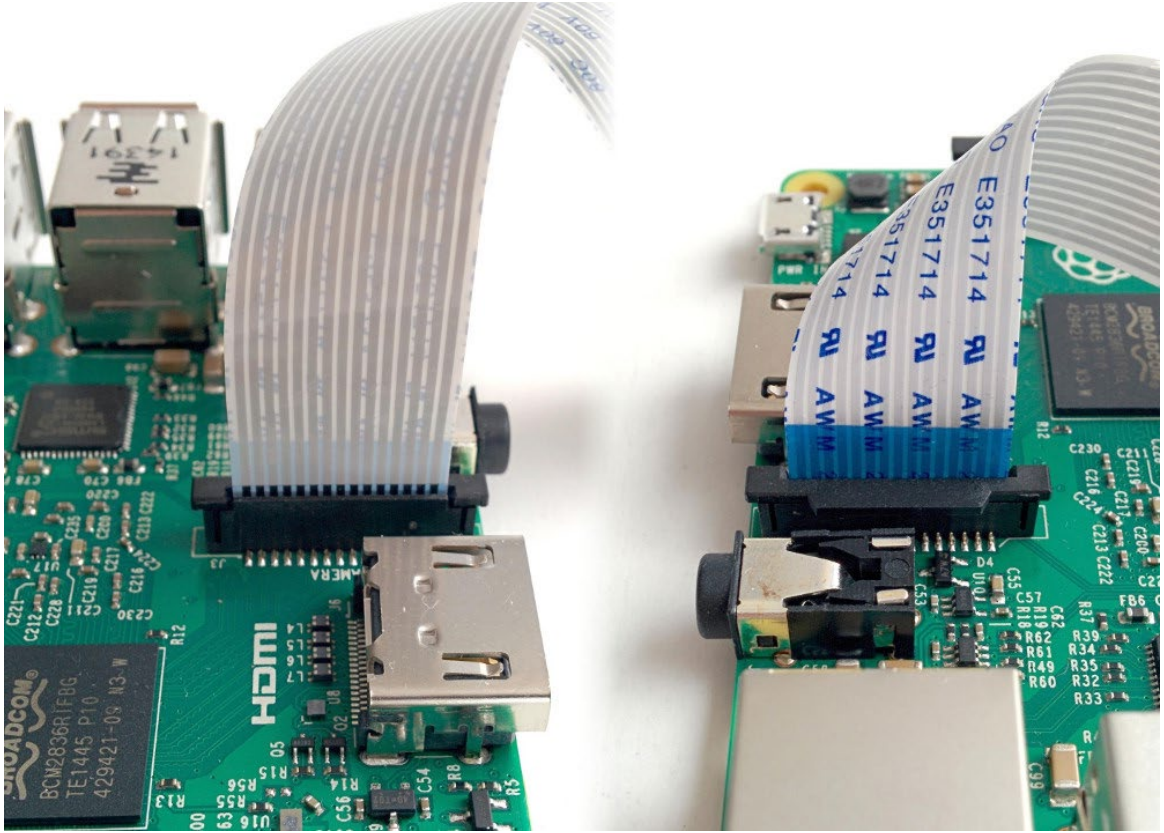





Raspberry Pi camera module



- 5Mpx / 8Mpx
- Full HD
- Photo & video
- Command line
- Python module
- Infra-red camera

Connect the camera



What does it add?

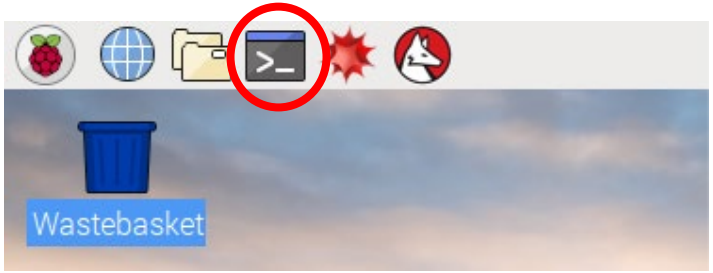
Timelapse

High Speed recording

Sensor & Images

Infrared photography

Test your camera

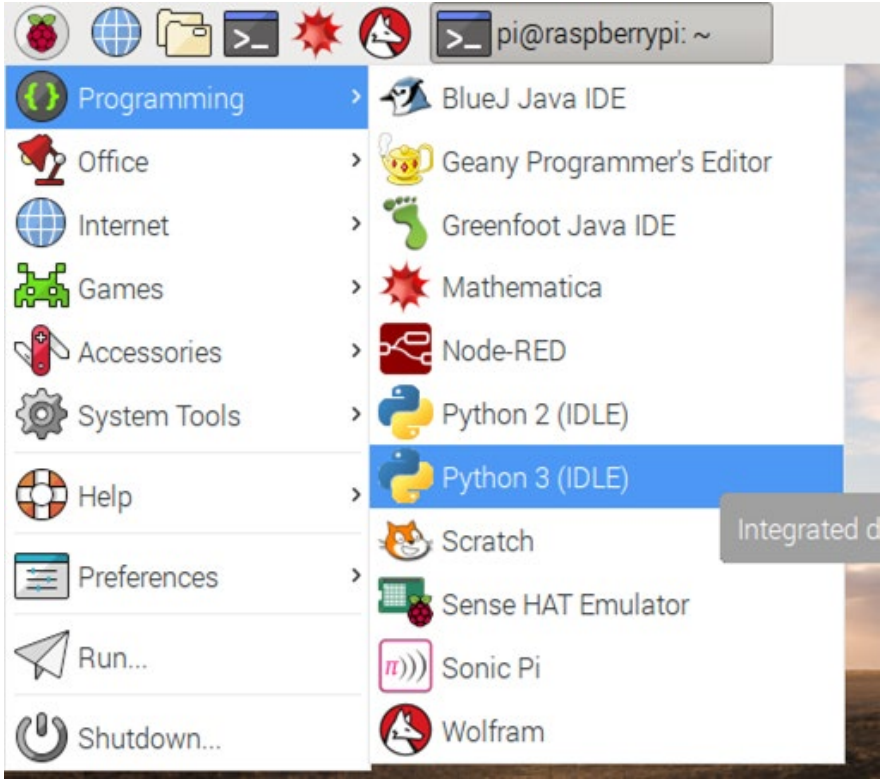



```
pi@raspberrypi:~ $ raspistill -k
```

Ctrl + C to close preview

```
pi@raspberrypi:~ $ raspistill -o image.jpg
```

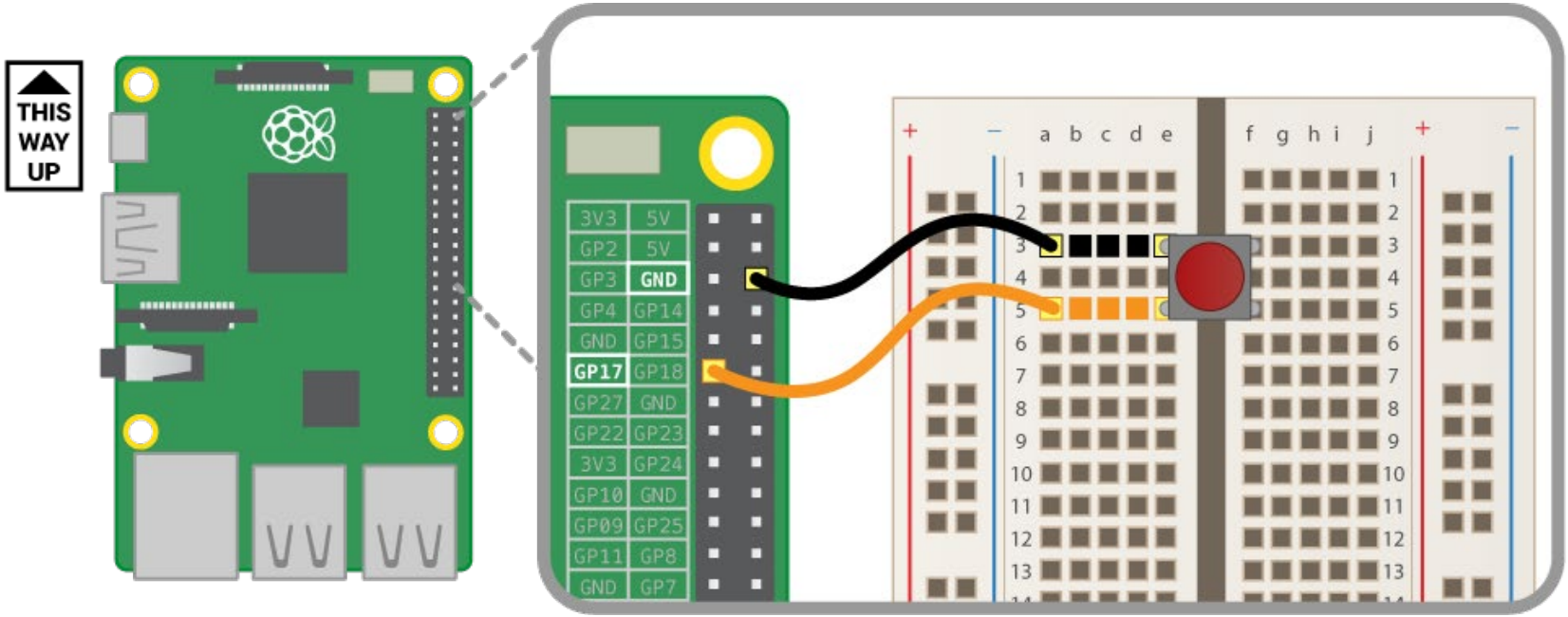
Create New Python 3 File



Take a selfie

```
## selfie.py
from picamera import PiCamera
from time import sleep

camera = PiCamera()

camera.start_preview(alpha=192)
sleep(3)
camera.capture("/home/pi/image.jpg")
camera.stop_preview()
```


Add a GPIO Button



Add GPIO Button code

```
## button.py
from picamera import PiCamera
from gpiozero import Button
from time import sleep

camera = PiCamera()
button = Button(17)

camera.start_preview(alpha=192)
button.wait_for_press()
sleep(3)
camera.capture("/home/pi/button.jpg")
camera.stop_preview()
```

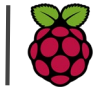

Add a loop

```
## loop.py
from picamera import PiCamera
from gpiozero import Button
from time import sleep

camera = PiCamera()
button = Button(17)

camera.start_preview(alpha=192)
for i in range(5):
    button.wait_for_press()
    sleep(1)
    camera.capture("/home/pi/button{0}.jpg".format(i))
camera.stop_preview()
```

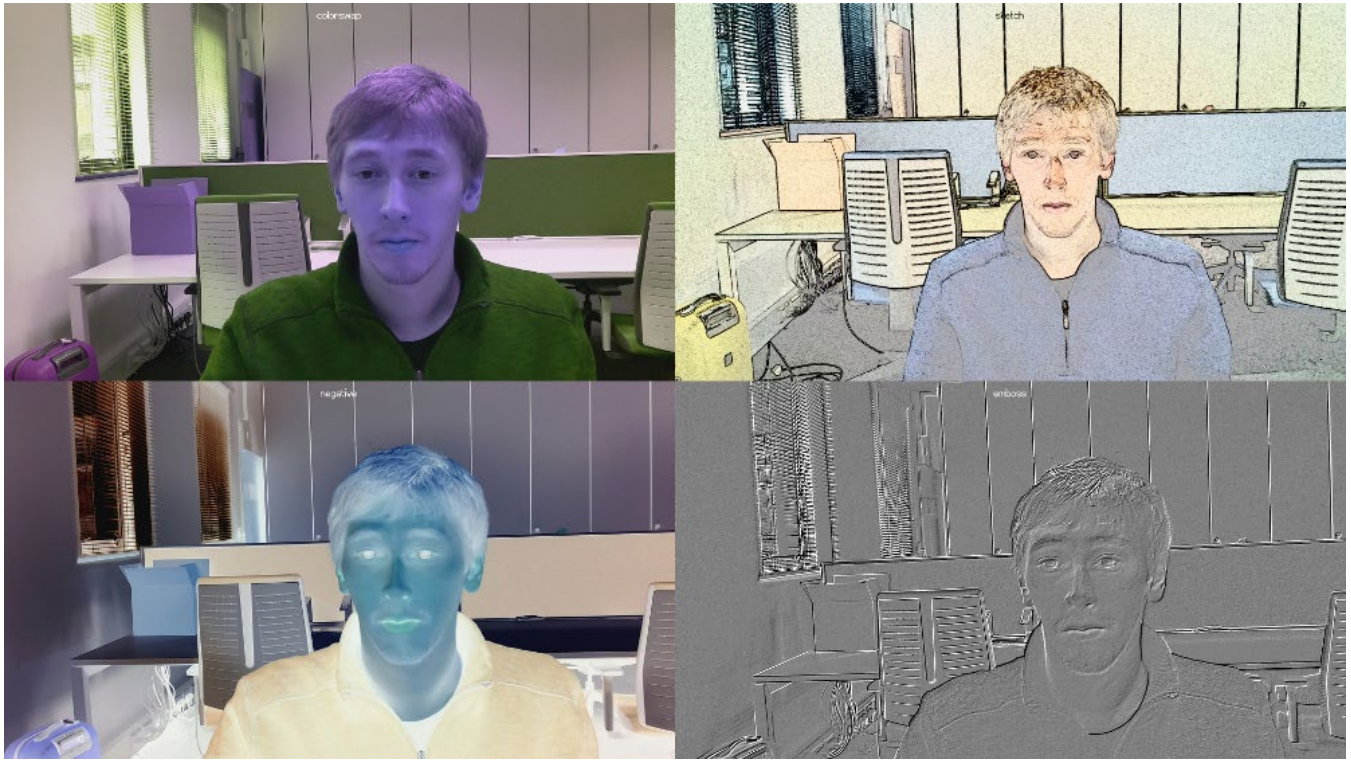

What's the difference?

```
…

for i in range(5):
    button.wait_for_press()
    sleep(3)
    camera.capture("/home/pi/button{0}.jpg".format(i))
```

```
…

button.wait_for_press()
for i in range(5):
    sleep(3)
    camera.capture("/home/pi/button{0}.jpg".format(i))
```

Picamera effects



Picamera effects

```
...

camera.start_preview(alpha=192)
button.wait_for_press()
camera.image_effect = 'negative'
sleep(5)
camera.capture("/home/pi/negative.jpg")
camera.stop_preview()
```

1. Start with **selfie.py**
2. Save As **effect.py**

Try more effects:

- negative
- colorswap
- sketch
- emboss

Capturing video

```
##video.py
from picamera import PiCamera

camera = PiCamera()
camera.start_preview(alpha=192)
camera.framerate = 24
camera.start_recording('my_video.h264')
camera.wait_recording(15)
camera.stop_recording()
camera.stop_preview()
```


Play your video

Open a Terminal window (Ctrl+Alt+T)

```
pi@raspberrypi:~ $ omxplayer myvideo.h264
```


Button → Motion Sensor

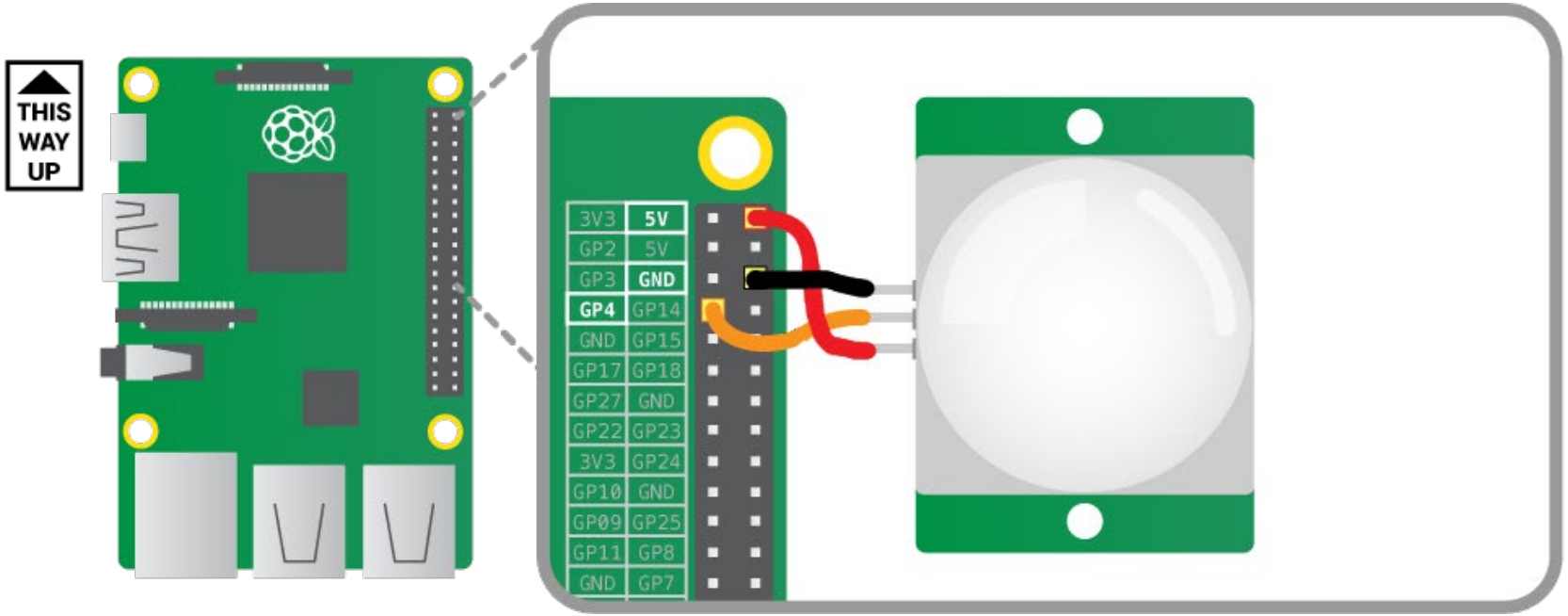
```
# button.py
from picamera import PiCamera
from gpiozero import Button
from time import sleep

camera = PiCamera()
button = Button(17)

camera.start_preview(alpha=192)
button.wait_for_press()
sleep(3)
camera.capture("/home/pi/button.jpg"
)
camera.stop_preview()
```

```
## motion.py
from picamera import PiCamera
from gpiozero import MotionSensor
from time import sleep

camera = PiCamera()
sensor = MotionSensor(4)

camera.start_preview(alpha=192)
sensor.wait_for_motion()
sleep(3)
camera.capture("/home/pi/pir.jpg"
)
camera.stop_preview()
```

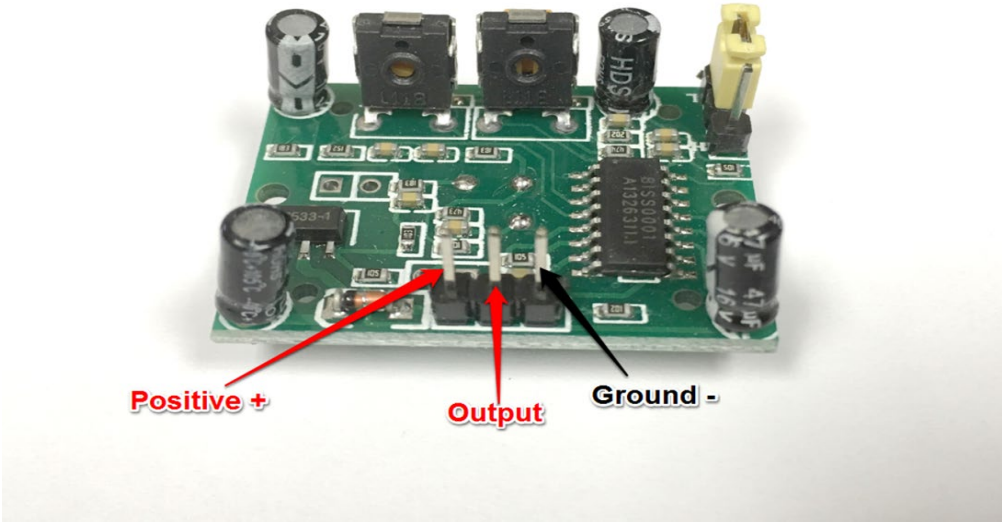

Capturing a time lapse

```
##timelapse.py
from picamera import PiCamera
from time import sleep

camera = PiCamera()
for num in range(1440):
    sleep(60)
    camera.start_preview(alpha=192)
    camera.capture("/home/pi/timelapse{0}.jpg".format(num))
    camera.stop_preview()
```


Connect a PIR Motion Sensor



Back of sensor



Documentation and help guides

picamera.readthedocs.io

gpiozero.readthedocs.io

raspberrypi.org/resources

raspberrypi.org/education/downloads